August 1986 UILU-ENG-86-2226

COORDINATED SCIENCE LABORATORY
College of Engineering

FOURIER DOMAIN INTERPOLATION TECHNIQUES FOR SYNTHETIC APERTURE RADAR

Bruce Clatworthy Mather

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY: N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release, distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): UILU-ENG-86-2226	5. MONITORING ORGANIZATION REPORT NUMBER(S): N/A
6a. NAME OF PERFORMING ORGANIZATION: Coordinated Science Laboratory, Univ. of Illinois; 6b. OFFICE SYMBOL (If applicable): N/A	7a. NAME OF MONITORING ORGANIZATION: Lockheed Missiles and Space Co.
6c. ADDRESS (City, State and ZIP Code): 1101 W. Springfield Avenue, Urbana, Illinois 61801	7b. ADDRESS (City, State and ZIP Code): Palo Alto, CA 94304
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: U. S. Air Force; 8b. OFFICE SYMBOL (If applicable): N/A	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER: Contract #F33615-83-C-1034
8c. ADDRESS (City, State and ZIP Code): WPAFB, Dayton, OH 45433	10. SOURCE OF FUNDING NOS. — PROGRAM ELEMENT NO.: N/A; PROJECT NO.: N/A; TASK NO.: N/A; WORK UNIT NO.: N/A
11. TITLE (Include Security Classification): Fourier Domain Interpolation Techniques for Synthetic Aperture Radar	
12. PERSONAL AUTHOR(S): Mather, Bruce C.	
13a. TYPE OF REPORT: Technical; 13b. TIME COVERED: FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): August 1986; 15. PAGE COUNT: 161
16. SUPPLEMENTARY NOTATION: N/A	

17. COSATI CODES		
FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number): Synthetic aperture radar, radar imaging, multidimensional interpolation.

19. ABSTRACT (Continue on reverse if necessary and identify by block number): Spotlight-mode synthetic aperture radar (SAR) produces complex Fourier data points on a polar grid which is offset from dc in the frequency domain. To produce an image in the spatial domain, it is necessary to invert this sampled Fourier data prior to extracting magnitude information. However, the polar format of the data makes this difficult, since there is no known polar FFT. An alternative is to interpolate the complex polar data to a Cartesian grid and then perform the two-dimensional FFT. The magnitude of the resulting data array represents the magnitude of the complex ground reflectivity of the terrain under illumination. The interpolation process can be very computationally intense, with an order two to fifty times that of the FFT. Reducing the computation in the interpolation stage, while maintaining reconstruction quality is the focus of this work. Several 2D interpolation techniques are examined, including nearest neighbor, bilinear, inverse-distance to the nth power, weighted sinc, chirp z-transform, and the newest interpolation algorithm proposed for this problem - the cubic spline. It is found that separable interpolation schemes outperform the more commonly used nearest neighbor and inverse distance algorithms,

(Continued on reverse)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER (Include Area Code); 22c. OFFICE SYMBOL: None

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

19. ABSTRACT (cont.)
and that the cubic spline is very competitive the weighted sinc interpolator in computation requirements and reconstruction quality. The chirp z-transform is determined to be a good alternative to the classical interpolation-DFT approach.

FOURIER DOMAIN INTERPOLATION TECHNIQUES FOR SYNTHETIC APERTURE RADAR

BY

BRUCE CLATWORTHY MATHER

B.S., University of Illinois, 1980
M.S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1986

Urbana, Illinois

FOURIER DOMAIN INTERPOLATION TECHNIQUES FOR SYNTHETIC APERTURE RADAR

Bruce Clatworthy Mather, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1986

Spotlight-mode synthetic aperture radar (SAR) produces complex Fourier data points on a polar grid which is offset from dc in the frequency domain. To produce an image in the spatial domain, it is necessary to invert this sampled Fourier data prior to extracting magnitude information. However, the polar format of the data makes this difficult, since there is no known polar FFT. An alternative is to interpolate the complex polar data to a Cartesian grid and then perform the two-dimensional FFT. The magnitude of the resulting data array represents the magnitude of the complex ground reflectivity of the terrain under illumination. The interpolation process can be very computationally intense, with an order two to fifty times that of the FFT. Reducing the computation in the interpolation stage, while maintaining reconstruction quality is the focus of this work. Several 2D interpolation techniques are examined, including nearest neighbor, bilinear, inverse-distance to the nth power, weighted sinc, chirp z-transform, and the newest interpolation algorithm proposed for this problem - the cubic spline. It is found that separable interpolation schemes outperform the more commonly used nearest neighbor and inverse distance algorithms, and that the cubic spline is very competitive the weighted sinc interpolator in computation requirements and reconstruction quality. The chirp z-transform is determined to be a good alternative to the classical interpolation-DFT approach.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Kenneth Jenkins for his continued support of my work and encouragement. He often had more belief in me than I did and gave me much moral support in the final hours. I would also like to thank Professor Dave Munson for allowing me to bother him with all my plots and ideas. Acknowledgements are due to the additional committee members Professors W. D. O'Brien, Jr. and T. S. Huang, and to Professor Richard Brown for exposing me to the instructional side of academia. Special thanks to Michael Haney for listening to my lamentations, ideas, and thoughts and giving me encouragement and more ideas to consider. Thanks also to Phill West for providing me with his ear and to Eric Diethorn for putting up with me, and to Jim Krogmeier for putting up with my computer. Chuck and Diane get the "best friends" award for their constant encouragement, love, and companionship. This *is* the last time. Warm thanks are also due my parents, whom without, I would not have been. A loving thank you to my wife, Victoria, for bearing with me through these seemingly endless years and having the strength to force me to finish by stepping out on her own. Finally, I must thank God for giving me the patience and skills to accomplish this task. May it be to His glory.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Radar, or *Ra*dio *D*etecting *a*nd *R*anging, has been used for decades to detect distant objects, measure their velocity, and more recently, for terrain mapping. It was theorized in the early 1900s, and first demonstrated in the mid-30s. Subsequently, it was developed to detect airplane bombers during WWII [1]. Since then, it has been refined and used in hundreds of applications such as space exploration, airplane avoidance systems, insect and bird tracking, weather observation and prediction, and, of course, highway speed enforcement.

When radar is used to simply detect the range to a distant object, a series of electromagnetic pulses is transmitted in the direction of the object and the two-way time delay of the reflected pulses is measured. Knowing the pulses propagate at the speed of light, we can readily calculate the range to the object. If we wish to measure the radial velocity of an object, we can transmit a continuous wave (CW) electromagnetic signal (a simple sinusoid) and measure the shift in frequency of the returned signal induced by the moving object. Again, it is simple to calculate the velocity of the object from the frequency shift (Doppler shift) and the known propagation velocity of the signal.

When the azimuthal position (perpendicular to range) is also desired, the transmitted signal must be sent as a narrow beam which is no wider than the desired azimuthal resolution. It is easily shown that at a range R and carrier frequency ω_c, an antenna with an aperture size D will have an azimuthal resolution of order $R\omega_c/D$ meters. For radar operating at conventional microwave frequencies (10^9 Hz), this implies that very small resolution requirements dictate a physical antenna of gigantic proportions - several thousand feet across! For an airborne antenna, this is impractical; however, very high azimuthal resolution can be achieved by *synthesizing* a large phased antenna array in an imaging system called synthetic aperture radar.

Synthetic aperture radar (SAR) is a means of obtaining high resolution terrain maps (complex reflectivity maps) by coherently processing the backscattered radar signal's phase (referred

to as "phase history "). In 1953, the University of Illinois experimentally demonstrated strip mapped SAR[2]. This was the earliest form of SAR in which the airborne radar antenna is held at a fixed squint angle relative to the terrain patch of interest. Coherent processing of the returned radar signal produces an effective aperture many times the size of the physical antenna, thus creating an azimuthal beamwidth proportionally narrower, resulting in finer azimuthal resolution.

A higher resolution form of SAR is called spotlight mode SAR. This mode requires that the antenna be continuously positioned, or steered, toward the center of the terrain patch during the plane's flight. The same patch is effectively pictured from many different angles, and when coherently processed, produces a much higher resolution image than could be obtained from a single observation angle.

Initially, SAR reflectivity data were processed optically. The phase histories were recorded on film[1] and then processed on an optical bench through a complex and bulky arrangement of lenses and coherent light [3, 4]. The data were continuous in the range direction and sampled in the azimuth coordinate (the discrete return pulses). An optical Fourier transform is thus continuous in range and discrete in azimuth and computed almost instantaneously. Although high resolution was achieved, the characteristics of the optical processing equipment made the reconstruction procedure cumbersome in the laboratory and impossible during inflight data collection. The advent of very fast digital processors and inexpensive semiconductor memory made real-time imaging feasible, i.e., the terrain maps could be (theoretically) generated in the data gathering platform online and displayed during the flight. Digital processing of the signals led to other problems, however, and such issues as polar sampling criterion and polar-to-digital interpolation became the focus of much study. Stark [5] presents some polar sampling theorems for the case of complete 2π sampling rasters such as those used in computer-aided tomography (CAT); however, they are difficult to apply to the case of spotlight mode SAR

[1]A light source, modulated by the return signal, is mechanically moved over film at a speed proportional to the sweep rate of the chirp radar.

where the recorded data region is a small slice of a toroid instead of the complete disk shaped region of CAT.

In the polar format algorithm of recording the phase histories of the SAR data, the sampled data (after coherent demodulation) are placed on a small section of a polar grid. If an inverse discrete Fourier transform (IDFT) for polar grids were available, then the radar data could theoretically be inverse-transformed directly and the final image displayed on a polar display device. Such an IDFT is not known, however, and the data must be interpolated to a rectangular grid before the IDFT operation. Even if such an IDFT were possible, polar displays are not readily available (though work is being done in this area [6]), and so interpolation in the spatial domain would be required for display on a rectangular raster. There already exist several fast Fourier transform (FFT) algorithms as well as special purpose FFT hardware available to perform the IDFT for rectangular sampled data arrays.

The most computationally intense task in generating the radar image involves two-dimensional interpolation from the polar grid to the rectangular grid. While an N by N 2-D FFT requires $O(N^2 \log_2 N)$ operations, the 2-D interpolation step often requires $O(K*N^2)$ where K is a number ranging from $\log_2 N$ to as high as N^4, depending on the interpolation algorithm and order. Thus, the interpolation step enormously overshadows the FFT step in computational complexity, i.e., processing time. This makes the SAR imaging tool difficult to implement in real time.

The interpolation requirements of SAR are very similar to those of the direct Fourier reconstruction techniques used in CAT. Jenkins, Munson and O'Brien[7] have developed an analogy between SAR and CAT which places projection data on the polar grid. There are two main differences between SAR and CAT:

(1) SAR uses coherent imaging and retains the phase information from its complex targets while CAT is non-coherent (this is not true of diffraction tomography which *is* a coherent system that records magnitude *and* phase) and thus only the magnitude of the projection

is recorded. The recorded CAT data are also strictly positive, while SAR data are complex and may have negative real and imaginary parts.

(2) CAT Fourier data lie in a polar grid which usually spans the full 2π in angle, and radially from 0 to some r_{max}, while SAR data lie in a polar annulus whose width is proportional to the radar signal bandwidth, and which is far removed from the origin occupying only a few degrees of angular width (typically 2 to 10 degrees).

For typical systems, the geometry of the SAR data grid is very close to a rectangular, or a rotated rectangular grid. The polar grid in CAT does not share this property. Given the small data record of collected phase histories, and the fact that the recorded Fourier section is offset from the Fourier origin (no dc in azimuth), it is surprising that one can obtain any azimuthal resolution at all. Munson and Sanz [8] have demonstrated that it is the coherency of the SAR system, together with the complex nature of the "spatial" targets, which allows such high image quality from such a limited amount of Fourier data. Despite these major differences between SAR and CAT imaging, many of the digital processing techniques from CAT can be applied to SAR. Modifications to these algorithms are made in accordance to the geometric difference between them and some approximations can be made in SAR which are inappropriate for tomography, but the basic results of sampling and windowing, as well as reconstruction, and error analysis of the computed tomography problem are applicable to SAR.

The reason that the interpolation requirements are so severe is due to the nature of the transform operation. Every point in the Fourier domain contributes to the spatial reconstruction. Single errors in the Fourier domain, therefore affect the entire spatial domain. It is very important, then, for the interpolator to be as accurate as possible, especially in the Fourier areas where most of the energy is concentrated. Without *a priori* information about the spectral energy distribution, it is thus imperative to be uniformly accurate throughout the region We shall see that there is a trade-off between interpolator accuracy (and thus image resolution) and computation. An important part of this research is dedicated to reducing the latter two while

maintaining the former.

Multidimensional interpolation is also important in other problems where the data gathering methods provide non-rectangular sampling geometries, e.g., underwater mapping [9], seismic mapping [9], imaging the ocean and sea-ice [10], and three-dimensional medical imaging [11, 12, 13] all gather data on sampling grids which may not be rectangular. Digital rotation, magnification, and reduction of images may also require two-dimensional interpolation.

The focus of this thesis is on the effects that various two dimensional interpolation schemes in the Fourier domain SAR data have on spatial domain reconstructions. A review of the fundamental spotlight mode SAR equations is presented in Chapter 2. The tomographic formulation of SAR is also reviewed and compared to the radar Doppler interpretation. Alternate inversion techniques are also briefly mentioned. A more detailed discussion of the interpolation problem in given in Chapter 3. Classical techniques are reviewed and extended to the SAR geometry. Spline interpolants are then presented from a DSP point of view, and a clarification of the term *spline* is given. The topics of windows, aliasing, spurious targets and artifacts are discussed in Chapter 4, as well the formulation of alternate sampling rasters. A rather lengthy set of empirical results for many different interpolator algorithms and sampling strategies is given in Chapter 5. It was felt that more insight could be obtained by studying these reconstructions rather than by producing complicated expressions for the spatially varying polar interpolators. Finally, a summary of the ideas and results presented in this work is presented in Chapter 6. Topics of further research to be done in the area of digital SAR image reconstruction is also discussed.

CHAPTER 2

SYNTHETIC APERTURE RADAR: THEORY AND BACKGROUND

In order to understand the interpolation problem, it is useful to see how the specific geometry developed. In the following sections, the SAR equations will be reviewed and the polar/rectangular geometry will be studied. The relationship to tomography will also be examined.

2.1 Derivation of Exact Point Target Response

Schwartz [14] investigated three data formats for SAR and included much of the analysis that was developed from Weis and Jenkins at Lockheed [15]. The analysis that follows was the basis for the original SAR simulation program, though it was later simplified to remove the quadratic phase term that appears in the phase equations. The signal response phase equations here shall be referred to as the *exact point target response*.

The coordinate system used is shown in Fig. 2.1. The data gathering platform moves with velocity V in the -x direction and at an altitude h. The ideal point target, P, is located at (x_t,y_t) which is measured relative to the reference point Q, located at (x_0,Y_0)[2]. Q is located at (0,0) on the stationary axes x'—y'. The terrain patch is assumed to be a square of size L by L. Note that the position of Q changes from pulse to pulse, since the reference axes (x-y) are moving relative to Q. The range variables r_{0n} and r_n are the distances to Q and P, respectively, and θ_n is the squint angle in the ground plane during the nth pulse. The subscript n is used to describe variables which change from pulse to pulse and so indicates the variable during the nth pulse when the look angle is θ_n. Later, n will be dropped, so that each variable is defined for an arbitrary angle θ.

[2]It is assumed that the distance from the data platform to the reference Q is known during the entire data collection window. Errors in this distance measure can defocus the reconstructed image, but can be corrected through a procedure known as autofocus [16].

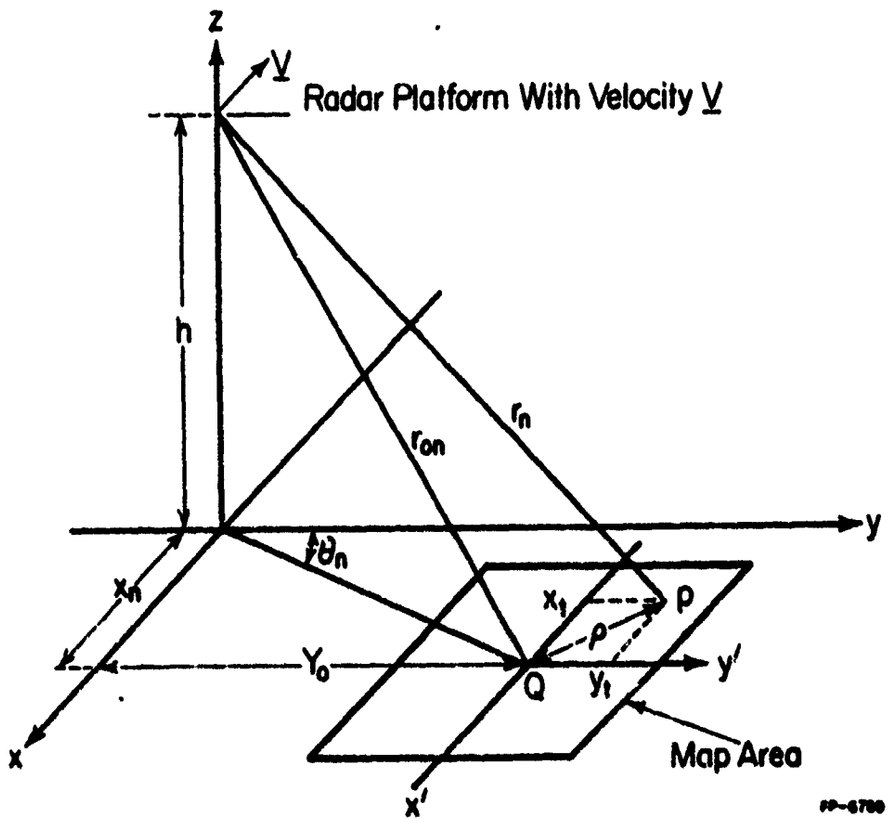


Figure 2.1 Coordinate system for Exact Point Target Response Derivation

The transmitted pulse is a high bandwidth chirp signal, i.e., a linearly modulated sinusoid:

$$s(t) = \begin{cases} \cos(\omega_c t + \frac{\nu t^2}{2}) & |t| \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where ω_c is the carrier frequency, ν is the linear FM sweep rate and T is the pulse duration.

The nth received pulse is

$$g_n(t) = \begin{cases} A\cos(\omega_c(t-\tau_n) + \frac{\nu}{2}(t-\tau_n)^2) & |t-\tau_n| < T/2 \\ 0 & \text{otherwise} \end{cases} \quad (2.2)$$

where A is the free space propagation attenuation factor and τ_n is the two-way travel time of the nth pulse. If $Y_0 >> L$, the propagation attenuation of the signal will be approximately the same over all the pulses, A will be set to unity. Note the quadratic phase term in (2.2). This will create problems later on. Note also that this particular analysis assumes that the aircraft position is essentially constant during each pulse. The analysis of Munson *et al.* [7] showed that the aircraft can have an instantaneous velocity of zero during each pulse, and that (classical) Doppler processing is therefore not strictly required for SAR.

Now, we apply stretch processing, i.e., pulse compression[3], to the returned signal. This is the technique of compressing the chirp pulse (and thereby increasing the range resolution of the radar) by coherently mixing the return signal with a simulated return from the reference target. Typically, this signal is generated by the transmitter sweep oscillator and delayed according to the distance to the known reference. If the distance to the reference is not known precisely, the simulated reference chirp signal will also be in error and will cause defocusing of the image, i.e., point targets will become smeared. The nth pulse reflected from the reference Q will have the form

$$g_{0n}(t) = \begin{cases} A\cos(\omega_c(t-\tau_{0n}) + \frac{\nu}{2}(t-\tau_{0n})^2) & |t-\tau_{0n}| < T_e/2 \\ 0 & 0 \end{cases} \quad (2.3)$$

where T_e is the effective pulse width determined by the size of the terrain being imaged.

Multiplying $g_n(t)$ and $g_{0n}(t)$, and low pass filtering (to retain only the difference frequency), we obtain the phase function Φ:

[3] Stretch processing and pulse compression have seemingly opposite connotations, but refer to the same technique of processing a *stretched pulse* and compressing the pulse-width through autocorrelation of a well-designed pulse signal, e.g., a linearly modulated chirp pulse. See Klauder et al. [17] for more information on chirp radars.

$$\text{Lowpass Filter}\{\, g_n(t) \cdot g_{0n}(t) \,\} = \cos(\Phi_n(t)) \tag{2.4}$$

where

$$\Phi_n(t) = \omega_c(t-\tau_n) + \frac{\nu}{2}(t-\tau_{0n})^2 - \omega_c(t-\tau_{0n}) - \frac{\nu}{2}(t-\tau_n)^2 \tag{2.5a}$$

$$= (\tau_n-\tau_{0n})(\omega_c + \nu t) - \frac{\nu}{2}(\tau_n^2-\tau_{0n}^2) \tag{2.5b}$$

This phase function corresponds to the I (in-phase) channel of the demodulator (mixer and low-pass filter). To obtain the Q channel (quadrature) output, multiply the return signal by a form of (2.3) with *cos* replaced by *sin* and then low pass filter to retain the difference frequency.

$$\text{I channel output: Real}\{e^{j\Phi_n(t)}\} = \cos(\Phi_n(t))$$

$$\text{Q channel output: Imag}\{e^{j\Phi_n(t)}\} = \sin(\Phi_n(t))$$

We would like to now convert the ensemble of return signals into a two-dimensional function and show that it is an approximation to the phase of the 2-D Fourier transform of a point target. The two-way time delay τ to a target at range r is 2r/c, where c is the propagation velocity of the radar signal in free space. Substituting for τ in (2.5b), the phase function becomes

$$\Phi_n(t) = \frac{2}{c}(r_n-r_{0n})(\omega_c + \nu t) - \frac{2\nu}{c^2}(r_n^2-r_{0n}^2) \tag{2.6}$$

It is now useful to make a change in variables

$$z_n = (r_n/r_{0n})^2 - 1 \tag{2.7}$$

Note that for our geometries, $|z_n| << 1$. Applying the Pythagorean theorem to Fig. 2.1 and making the substitution $x_n = Y_0 \tan\theta_n$ we obtain

$$\left(\frac{r_n}{r_{0n}}\right)^2 = \frac{(x_n + x_t)^2 + (Y_0 + y_t)^2 + h^2}{x_n^2 + Y_0^2 + h^2} \tag{2.8}$$

$$\left|\frac{r_n}{r_{0n}}\right|^2 = 1 + \frac{2\,x_n\,x_t + 2\,Y_0\,y_t + x_t^2 + y_t^2}{r_{0n}^2} \tag{2.9}$$

$$z_n = \frac{2\,Y_0\,x_t\cot\theta_n + 2\,Y_0\,y_t + x_t^2 + y_t^2}{r_{0n}^2} \tag{2.10}$$

The change of variables equations in (2.7) can be rearranged to obtain

$$r_n - r_{0n} = r_{0n}(\sqrt{1 + z_n} - 1) \tag{2.11a}$$

and

$$r_n^2 - r_{0n}^2 = r_{0n}^2\,z_n \tag{2.11b}$$

Since $|z_n| << 1$, the term $\sqrt{1 + z_n}$ can be expanded into a power series

$$\sqrt{1 + z_n} = 1 + \frac{z_n}{2} + \frac{z_n^2}{8} + \frac{z_n^3}{16} + \cdots$$

in which only the first 2 terms are retained. Thus

$$r_n - r_{0n} \approx \frac{r_{0n}\,z_n}{2} \tag{2.12}$$

Substituting (2.12) into (2.6) results in an approximate phase function

$$\Phi_n(t) = \frac{r_{0n}\,z_n}{2}\left|\omega_c + \nu t - \frac{2\nu r_{0n}}{c}\right| \tag{2.13}$$

Note that r_{0n} and z_n are each functions of n, as well as θ_n, the angle at which the nth pulse is transmitted. The notation can be modified to express these variables simply as functions of θ. This results in the specification of $\Phi_n(t)$ as a function of t and θ.

$$\Phi(t,\theta) = \frac{r_0(\theta)\,z(\theta)}{2}\left|\omega_c + \nu t - \frac{2\nu r_0(\theta)}{c}\right| \tag{2.14}$$

Now, Φ is a two-dimensional function which can be mapped as a function of θ and t. Note that r_0 and z are explicitly shown as functions of θ. This notation is simply to emphasize this functional relationship.

The variable t can be functionally translated into a range variable by defining R as a function of time

$$R(t) = K_v\, t + K_{offset} \quad . \tag{2.15}$$

where K_v represents the scaling in the radial (time) direction and K_{offset} is the offset from the origin from which valid received data begin[4]. Solving (2.15) for t and substituting the resulting expression into (2.14) yields an expression for Φ in terms of R and θ, which is a two-dimensional function of polar coordinates.

$$\Phi(R,\theta) = \frac{r_0(\theta)\, z(\theta)}{2} \left| \omega_c + \nu \frac{R - K_{offset}}{K_v} - \frac{2\nu r_0(\theta)}{c} \right| \tag{2.16}$$

A proper choice of the scaling factors K_v and K_{offset} will simplify (2.16). Let

$$\frac{K_{offset}}{K_v} = \frac{\omega_c}{\nu} - \frac{2r_0}{c} \tag{2.17}$$

and substitute (2.17) into (2.16):

$$\Phi(R,\theta) = \frac{\nu R}{cK_v} r_0(\theta) z(\theta) \tag{2.18}$$

Substituting (2.10) into (2.18) gives

$$\Phi(R,\theta) = \frac{2\nu Y_0 R}{cK_v r_0} \left(y_t + x_t \cot\theta + \frac{\rho^2}{2Y_0} \right) \tag{2.19}$$

where $\rho^2 = x_t^2 + y_t^2$, the squared radial distance of P to Q.

To eliminate K_v and make (2.19) into a more useful form, choose K_v such that

$$K_v = \frac{Y_0\, c}{2\, r_0 \sin\theta} \tag{2.20}$$

yielding

[4] The received signal in (2.2) is only valid during a small time interval which is dependent on the nearest and farthest points of the imaged terrain. The nearest (and farthest) point is that point which is the minimum (and maximum) distance from the data platform to the terrain patch over the entire data gathering interval.

$$\Phi(R,\theta) = \frac{\nu R}{c^2}(y_t \sin\theta + x_t \cos\theta + \frac{\rho^2 \sin\theta}{2Y_0}) \tag{2.21}$$

And in rectangular coordinates where U=R cos(θ) and V=R sin(θ)

$$\Phi(U,V) = \frac{\nu}{c^2}(x_t U + y_t V + \frac{\rho^2 V}{2y0}) \tag{2.22}$$

This has the same form as the phase function of the Fourier transform of an offset delta function. The difference between this expression and the ideal point target response is the addition of an extra term - the *quadratic phase term*. This quadratic term is dependent on ρ^2 which is the squared distance of the point target to the image center (reference). As the target is moved out radially from the center, the quadratic term grows and causes a noticeable smearing of the response, specifically, a point target widens. Compare this to the inverse Fourier transform, $\hat{F}$, of a point target at (x_t,y_t):

$$\hat{F}(U,V) = F^{-1}\{\delta(x_t,y_t)\} = \int_{-\infty}^{\infty} \delta(x_t,y_t)\, e^{j(xU + yV)}\, dxdy = e^{j(x_t U + y_t V)} \tag{2.23}$$

$$F(U,V) = e^{j\Psi\left|x_t U + y_t V + \frac{\rho^2 V}{2y0}\right|} \tag{2.24a}$$

$$\approx \hat{F}(U,V) \tag{2.24b}$$

where $\Psi = \frac{\nu}{c^2}$ has dimensions radians/meters^2.

Recall that this is only an approximation to the exact point target response since the high-order terms of z_n were ignored. The exact point target response can be formulated by substituting (2.11a), (2.11b), (2.14), (2.17), and (2.20) into (2.6):

$$\Phi(R,\theta) = \frac{2\nu r_0^2(\theta)}{c^2}\left|\left(\frac{2R\sin\theta}{Y_0}+2\right)\left(\sqrt{1+z(\theta)}-1\right)-z(\theta)\right| \tag{2.25a}$$

$$r_0^2(\theta) = Y_0^2 \csc^2\theta + h^2 \tag{2.25b}$$

$$z(\theta) = \frac{2Y_0(x_t\cot\theta + y_t) + x_t^2 + y_t^2}{r_0^2(\theta)} \tag{2.25c}$$

And in rectangular coordinates:

$$\Phi(U,V) = \frac{2\nu r_0^2(U,V)}{c^2}\left|\left(\frac{2V}{Y_0}+2\right)\left[\sqrt{1+z(U,V)}-1\right]-z(U,V)\right| \tag{2.26a}$$

$$r_0^2(U,V) = Y_0^2\left|1+\frac{U^2}{V^2}\right| + h^2 \tag{2.26b}$$

$$z(U,V) = \frac{2Y_0(x_t\frac{V}{U} + y_t) + x_t^2 + y_t^2}{r_0^2(U,V)} \tag{2.26c}$$

These equations relate the position of a point target at (x_t,y_t) to its continuous complex phase function $\Phi(U,V)$. The data which are obtained from the imaging system consist of samples of Φ on a polar grid.

The phase function relates the complex 2-D frequency pattern to the position of the point targets. This means that a straightforward spectral analysis of the ensemble phase function (though sampled) should lead directly to the positions of the point target. Superposition is also applicable here, so that a continuous complex reflectivity map can be generated from the Fourier transform of the reflected and processed pulses.

2.2 The Tomographic Formulation of SAR

Munson, O'Brien and Jenkins [7] have developed the theory of SAR from a tomographic point of view. They examine SAR as a narrow-band filtered tomography problem using the projection slice theorem. Bernfeld [18] later develops the same sort of link between the two areas, although many of the processing considerations seem to be lacking.

In the following development, spatial domain functions will be lower case, Fourier domain functions will be upper case and functions of polar coordinates will have the subscript

p; i.e., $f(x,y)$ is a spatial domain function in rectangular coordinates and $F_p(\rho,\theta)$ is a polar representation of the Fourier transform of $f(x,y)$. $F(\omega_x,\omega_y)$ is given by

$$F(\omega_x,\omega_y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\, e^{-j(x\omega_x + y\omega_y)} dx\, dy \tag{2.27}$$

and

$$f(x,y) = \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(\omega_x,\omega_y)\, e^{+j(x\omega_x + y\omega_y)} d\omega_x\, d\omega_y \tag{2.28}$$

A projection of a multidimensional signal is a new function with dimension one less than the original signal. In effect, one of the dimensions is *integrated out* via a line integral. The projection of $f(x,y)$ at an angle θ is given by

$$f(u;\theta) = \int_{-\infty}^{\infty} f(u\cos\theta - v\sin\theta, u\sin\theta + v\cos\theta) dv \tag{2.29}$$

A projection is a function of one variable, u, with the second parameter, θ, indicating the angle of the projection. The transform of $p(u;\theta)$ which is a one-dimensional transform with respect to u is

$$P(\omega;\theta) = \int_{-\infty}^{\infty} p(u;\theta) e^{-ju\omega} du \tag{2.30}$$

The projection slice theorem can thus be succinctly stated as

$$P(\omega;\theta) = F_p(\omega,\theta) \tag{2.31}$$

The proof is found in a variety of sources. This particular statement of the theorem is given in [13].

The projection slice theorem says that the Fourier transform of a projection of $f(x,y)$ at an angle θ is equal to a slice of $F(\omega_x,\omega_y)$ taken at the same angle θ and passing through the origin. The consequence is that the original function can be reconstructed by *filling in* the Fourier data space with the transforms of the projection data and then inverse transforming the result. The entire Fourier plane must be constructed with an infinite number of projections, since the

projection operation is performed at discrete angles.

In computed tomography, f(x,y) is an attenuation coefficient, or density function, which is to be constructed through non-invasive means. Collimated X-rays oriented at the projection angle θ perform the line integrals, i.e., projections, of f(x,y). Unfortunately, only a finite number of projections are taken, and each projection is sampled so that the resulting data set lies on a polar grid equally spaced in θ and R. This results in an ill-posed reconstruction problem [19]. Though the previous paragraph describes a method of reconstructing the original function f(x,y) from the sampled Fourier data (called *Direct Fourier Reconstruction*), there are other techniques such as Convolutional Back-Projection (CBP) and Algebraic Reconstruction Techniques (ART) which produce very good images. Current state-of-the-art systems have sub-millimeter resolution. Convolutional Back-Projection has the greatest favor in CAT reconstruction because it has the lowest computational burden for the required image quality.

Direct Fourier reconstruction suffers from the same geometric difficulties as the spotlight SAR problem. The output data set is presented on a polar grid and must be interpolated to a rectangular grid prior to the transformation step. Currently the computational requirements and resulting images are not competitive with CBP.

In the Munson-O'Brien-Jenkins formulation of SAR[7], the approach to the analysis is different from that of Weis[15]. It is simplified because the altitude of the aircraft is zero (though this is corrected later in the paper), and the ground patch is circular (leading to a simplified expression for determining what part of the return signal is valid). The simplifications, however, lead to great insight into the imaging process. The following analysis is taken from [7] and the geometry of the problem is in reference to Fig. 2.2, in which the radar is imaging a circular patch of radius l_c at a distance R from the patch center. The signal transmitted is as before in (2.1):

$$s(t) = \begin{cases} \cos(\omega_c t + \frac{\nu t^2}{2}) & |t| \leq \frac{T}{2} \\ 0 & \text{otherwise} \end{cases} \tag{2.32}$$

which can be thought of as the real part of a complex exponential

$$s(t) = \mathrm{Re}\left\{ e^{j(\omega_c t + \frac{\nu t^2}{2})} \right\} \tag{2.33}$$

The return signal from a small differential patch dx.dy centered at (x_0, Y_0) and having a complex reflectivity $g(x_0, Y_0)$ is

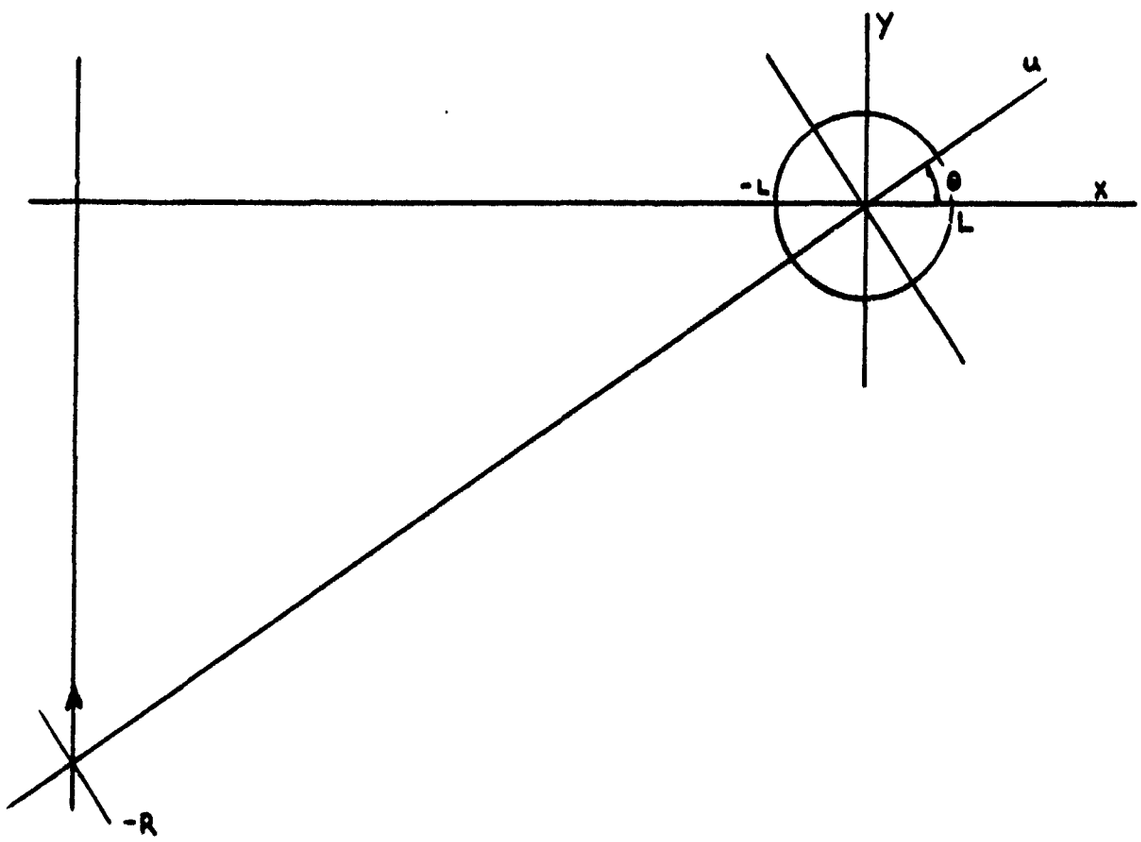


Figure 2.2 Geometry of Tomographic Derivation of SAR.

$$r_0 = A\,|g(x_0,Y_0)|\cdot\cos\left\{\omega_c(t-\frac{2R_0}{c}) + \frac{\nu}{2}(t-\frac{2R_0}{c})^2 + \arg\{g(x_0,Y_0)\}\right\} \qquad (2.34)$$

Again, just as in the previous analysis, the return signal contains a quadratic phase term. The attenuation constant A can again be set to unity since it will be approximately the same for all pulses if R>>L. Equation (2.34) can be simplified to

$$r_0 = \mathrm{Re}\left\{g(x_0,Y_0)\cdot s(t-\frac{2R_0}{c})dx\,dy\right\}. \qquad (2.35)$$

This is the return from a small differential element at a distance R_0 from the transmitter. For the geometry of Fig. 2.2, the locus of points at a distance R_0 from the data platform is an arc centered at the radar system. The return signal corresponding to the superposition of all differential patches at distance R_0 is the line integral along the same arc. However, if R>>L, then the circular (spherical) radar waves can be approximated by a plane wave normal to the u axis (the axis of transmission for an angle θ). This line integral is the value of the projection of g(x,y) onto the u-axis at u_0. Equation 2.35 can be thus rewritten as

$$\tilde{r}_0(t) = \mathrm{Re}\left\{p_\theta(u_0)\cdot s\left[t - \frac{2(R+u_0)}{c}\right]\right\}du \qquad (2.36)$$

where R_0 has been rewritten as R, the distance to the terrain center plus u_0. Note that these quantities are dependent on θ since R is changing as the plane flies past the patch. The return signal from the entire terrain patch is given by integrating $\tilde{r}_0(t)$ over the entire area (here the assumption that the terrain is circular simplifies the integration a great deal, though we could assume that a square patch is circumscribed by the circle, and the terrain between the circle and square has reflectivity zero.)

$$r'_\theta(t) = \mathrm{Re}\left\{\int_{-L}^{L} p_\theta(u)\,s\left[t - \frac{2(R+u)}{c}\right]du\right\} \qquad (2.37)$$

Because the transmitted signal is a chirp pulse, stretch processing is applied as before, which essentially translates positional information into frequency information. This is done by mixing (multiplying) this return r'_θ signal with a reference chirp from the ground patch center

$$\cos[\omega_c(t-\tau_0) + \frac{\nu}{2}(t-\tau_0)^2] \tag{2.38}$$

where $\tau_0 = \frac{2R}{c}$, the two-way travel time to the terrain center. This is low-pass filtered to obtain

$$r_\theta(t) = \frac{1}{2}\mathrm{Re}\left\{\int_{-L}^{L} p_\theta(u)\cdot\Psi(t)\cdot\Upsilon(t)du\right\} \tag{2.39}$$

where

$$\Psi(t) = e^{j\frac{2\nu u^2}{c^2}} \tag{2.40a}$$

is the quadratic phase term, and

$$\Upsilon(t) = \exp\left\{-j\frac{2}{c}(\omega_c + \frac{\nu}{2}(t-\tau_0))u\right\} \tag{2.40b}$$

The quadrature component is obtained by multiplying by the phase shifted reference chirp

$$\sin[\omega_c(t-\tau_0) + \frac{\nu}{2}(t-\tau_0)^2] \tag{2.41}$$

and low pass filter, giving the imaginary portion

$$r_\theta(t) = \frac{1}{2}\mathrm{Im}\left\{\int_{-L}^{L} p_\theta(u)\cdot\Psi(t)\cdot\Upsilon(t)du\right\} \tag{2.42}$$

of a complex signal

$$C_\theta(t) = \frac{1}{2}\left\{\int_{-L}^{L} p_\theta(u)\cdot\Psi(t)\cdot\Upsilon(t)du\right\} \tag{2.43}$$

If the quadratic phase term, $\Psi(t)$, can be removed, $C_\theta(t)$ resembles the Fourier transform of $p_\theta(t)$ in the u coordinate plane, evaluated at $[\omega_c + \frac{\nu}{2}(t-\tau_0)]$.

$$C'_\theta(t) = \frac{1}{2}P_\theta[\omega_c + \frac{\nu}{2}(t-\tau_0)] \tag{2.44}$$

If the transmitted chirp pulse had an infinite bandwidth, then $C'_\theta(t)$ would represent a continuous sample (slice) of the Fourier transform of the ground patch reflectivity, g(x,y). (Note that if the terrain reflectivity is isotropic i.e., g(x,y) is not a function of the radar look angle, then the projection is symmetric about the origin.) However, the pulse is only of finite width, thus finite bandwidth, and so the processed return represents only a small section of the Fourier slice. The return pulse is only valid for the interval

$$\frac{2(R+L)}{c} - \frac{T}{2} \leqslant t \leqslant \frac{2(R-L)}{c} + \frac{T}{2} \tag{2.45}$$

This interval represents the two-way time for the beginning of the pulse to travel to the end of the terrain patch until the end of the transmitted pulse reaches the beginning of the terrain patch. These calculations are based on minimum and maximum distances, which, for a rectangular patch, change from pulse to pulse (L would change non-linearly with θ). Again, the round patch has simplified the calculations. Substituting t of (2.45) into (2.44) yields an inequallity for the range of the argument of P_θ.

$$\frac{2}{c}(\omega_c - \nu T + \frac{2\nu L}{c}) \leqslant R_\omega \leqslant \frac{2}{c}(\omega_c + \nu T - \frac{2\nu L}{c}) \tag{2.46}$$

For a typical transmitted chirp pulse where $\omega_c \pm \nu T >> 2\nu L/c$, (2.44) reduces to

$$\frac{2}{c}(\omega_c - \frac{\nu T}{2}) \leqslant R_\omega \leqslant \frac{2}{c}(\omega_c + \frac{\nu T}{2}) \tag{2.47}$$

R_ω is bounded by the inner and outer radii of the slice of the Fourier transform of the terrain reflectivity represented by the processed return signal.

Thus, as before, the coherently processed signal is a section of a slice of the Fourier transform. This signal is also sampled uniformly in range so that the collection of sampled

returns assumes a polar grid, offset from the origin and having an angular sweep width of θ_M.

Many issues not covered, such as the quadratic phase term, modification for non-zero platform height, curvature of the radar signal and the effects of plane movement (inducing a Doppler term) are not included here, but can be found in[7]. As before, the quadratic phase term has the effect of smearing targets which are distant from the patch center. This is obvious, since the u^2 term represents the square of the radial distance of a target from the terrain origin. It is the same quadratic phase term found in the Weis analysis [15] and is difficult to remove during processing. It can be shown that this term limits system resolution and terrain patch size.

2.3 Comparison of CAT, SAR, and Analyses

Munson *et al.* [7] compare CAT and SAR and note important differences:

(1) As a result of stretch processing, the processed SAR signal is the Fourier transform of the projection. In CAT, the projection is still part of the spatial domain.

(2) The processed SAR signal gives only a small part of the projection transform devoid of dc components, which can result in the loss of edges and sections of constant reflectivity.

(3) In SAR, the projection is taken normal to the axis of imaging rather than parallel to the axis (as in CAT).

These two different ways of looking at the SAR problem lead to essentially the same result: the collected data are approximately samples of the Fourier transform of the terrain reflectivity function. With Doppler interpretation (Weis), the image is obtained by spectral estimation, i.e., a forward FFT, to translate sinusoids into discrete range bins. In the tomographic formulation, the data represent samples of the Fourier transform of the complex reflectivity which must be *inverse* transformed to obtain samples of the complex reflectivity. Both interpretations are correct, but can lead to confusion about the role of Doppler processing and the direction of the Fourier transform. In practice, it is of little concern, since the phase is

discarded in the final displayed image and forward or inverse transforms result in the same (though inverted and scale) magnitude map.

It should be pointed out that the Direct Fourier Reconstruction approach is but one algorithm for the terrain mapping. Earlier versions of strip mapped SAR, based on the same principle used the concept of multiple correlators to perform the stretch processing. The correlators performed the job of *matched filters*, which were tuned to point targets in each range bin, having the impulse response, which was the complex conjugate of the response of a point target of that range bin. This, however, involves much more processing than the FFT approach, though hardware implementations have been designed and built which can reduce the processing time through a pipelined architecture [20].

2.4 Other Inversion Techniques

The two-dimensional interpolation scheme is the most direct method for the SAR image reconstruction problem. Since a relationship was formed between SAR and CAT, it can be theorized that some of the tomographic reconstruction algorithms may be used in SAR. Unfortunately, many of them depend on the full set of projection data ($\theta_M = 2\pi$) and thus are difficult to apply to SAR. In particular, the algorithms that may be of use are convolutional back-projection (CBP) and the Hankel transform.

2.4.1 Convolution back projection and the Hankel transform

Quite a bit of research has been done in the area of tomographic reconstruction algorithms and their computational burden versus image quality (similar to the work presented here for SAR). A celebrated paper by Pan and Kak [21] details the tradeoffs of direct Fourier inversion with interpolation and CBP, but only very simple interpolators are used, notably the nearest neighbor and bilinear (used with a modification of the zero-padding-FFT technique described in Chapter 3). Interpolation via the circular sampling theorem (see Stark [5, 22] and Fan [23] for further discussion about this theorem) is presented, but again, this requires a periodic data set

(complete circular arcs) to be useful. Heffernan and Bates [24] also compared the very low-order interpolator methods, but applied them to the projection data prior to back-projection, rather than 2-D direct inversion. Similar work has been done by Mersereau [13]. Desai and Jenkins [25, 26] have researched the use of the CBP algorithm for the SAR problem and have achieved image quality comparable to the direct inversion technique, although at the expense of additional computation order. Even if the SAR problem could be considered a limited view tomography problem, the reconstruction techniques of bandlimited extrapolation to supply the rest of the Fourier data set would be far too costly for real time reconstruction.

The Hankel transform allows for direct polar-to-polar image reconstruction from the sampled Fourier data by expanding the two-dimensional image into a Fourier series and then performing a Hankel transform on these coefficients to obtain the Fourier series coefficients of the image. The use of the Hankel transform to invert the polar Fourier data set directly has been met with limited success [27] due to the cost of computing the needed transforms. Work is being done to develop a *Fast Hankel Transform* [28], but this transform still requires a full circular data set.

2.4.2 Spectral analysis of nonuniformly sampled data.

Hostetter [29, 30] developed a control systems approach to spectral observation of data which is irregularly sampled. The algorithm is not practical for large data records, however, because it is of order N^2 for N input points. For an N by N, 2-D array, this would require $O(N^4)$ calculations, which is not competitive with even the most sophisticated 2-D interpolator.

2.5 Polar-Rectangular Geometry for Interpolation

The first two sections of this chapter demonstrated how the spotlight mode SAR data set falls on a polar grid of limited size. The major thrust of the work here is to use the Direct Fourier Reconstruction algorithm to generate the complex reflectivity map of the imaged terrain and to study effects of different two-dimensional interpolators available. Fig. 2.3 details the

polar-to-rectangular grid relationships and the parameters needed to specify the relative geometry between the two rasters. The input polar grid is specified by

(1) R_{min} the inside radius of the torus of known Fourier data.

(2) θ_M, the maximum look angle of the SAR system.

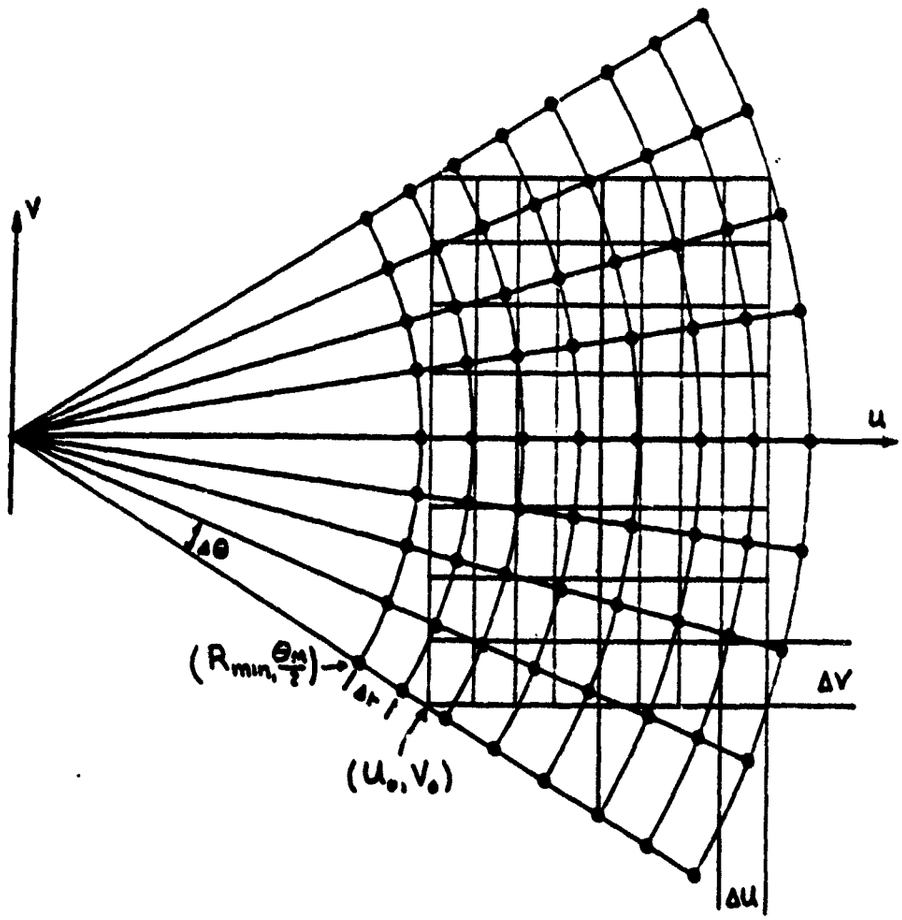


Figure 2.3 Polar-to-Rectangular Grid Geometries.

(3) Δr, the radial sampling increment.

(4) $\Delta\theta$ the angular sampling increment.

The output rectangular (square) raster is specified by

(1) u0,v0, the bottom-left coordinate of the square grid.

(2) $\Delta u, \Delta v$ the sampling increments for each of the u and v dimensions. Typically, these will be equal.

The resolution of the system is governed by the width of the Fourier section in each dimension. These widths, in turn, are determined by the radar carrier frequency ω_c, chirp sweep width, and the maximum look angle, θ_M. In the range direction, the width is $R_{max} - R_{min}$, the bounds on the range variable given above determined by the radar chirp sweep width. From (2.47) above, the resolution is given as approximately $\frac{2\nu T}{c}$. In the azimuthal dimension, the v-width is approximately that of the azimuthal extent of the annuluar section, approximately $4\omega_c \sin\theta_M$. If the processed response signal represented the Fourier space exactly (no quadratic phase term), then a point target placed at (x_0, Y_0) would yield a two-dimensional sinc due to the limited extent of the Fourier space available. The resolution of the system can be defined as the width of the sinc mainlobe divided by 2 since, theoretically, this represents the separation of two resolvable point targets. In practice, the resolution is degraded though constructive and destructive interference of the phase function.

To achieve maximum resolution, it is important to interpolate from the polar grid to the largest square which inscribes the angular section (geometry shown in Fig. 2.4). The range resolution is determined by the chirp sweep width (time-bandwidth product), and the azimuthal resolution is determined by the center frequency ω_c and the look angle θ_M. These parameters specify the annular section location and size in the Fourier domain. The parameters used in the simulations were ω_c and the look angle θ_M, from which the chirp sweep width and square's size and position are determined. The solution in this case is straightforward trigonometry.

Let W be the width of the square to be determined. R_{min} and R_{max} are also to be found - relating back the chirp sweep width of the radar.

$$q_0 = \frac{2\omega_c}{c} = \frac{R_{min} + R_{max}}{2} \tag{2.48}$$

Note that q_0 is exactly halfway between R_{min} and R_{max}, but is not exactly the square's center.

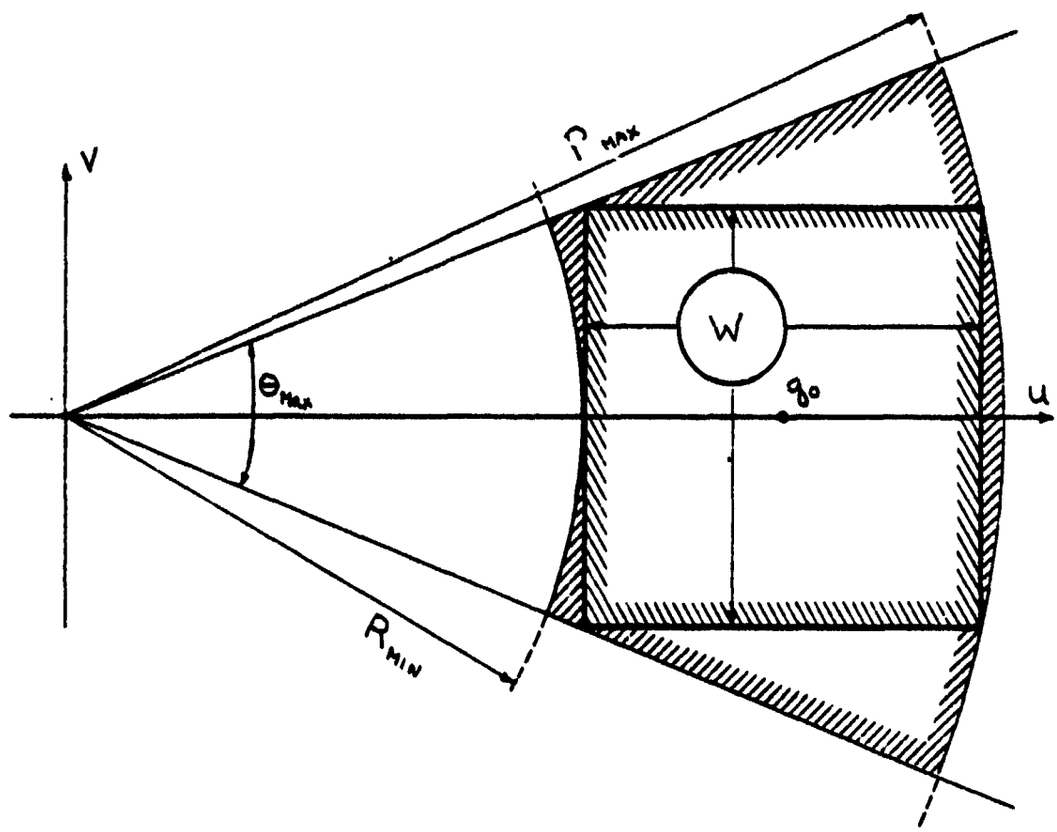


Figure 2.4 Determining the Largest Square Constrained by ω_c and θ_M.

There is a very slight gap between the right edge of the square and the torus on the v=0 axis. R_{max} is given by

$$R_{max} = \left[(r_{min} + W)^2 + (\frac{W}{2})^2\right]^2 \tag{2.49}$$

since the square must touch the outer radii for maximal size. W is related to R_{min} by

$$W = 2R_{min}\tan\frac{\theta_M}{2} \tag{2.50}$$

Substituting (2.50) into (2.49) gives

$$R_{max}^2 = (R_{min} + 2R_{min}\tan\frac{\theta_M}{2})^2 + (R_{min}\tan\frac{\theta_M}{2})^2. \tag{2.51}$$

But from (2.43), $R_{max} = 2q_0 - R_{min}$. Substituting this into (2.51) results in

$$(2q_0 - R_{min})^2 = (R_{min} + 2R_{min}\tan\frac{\theta_M}{2})^2 + (R_{min}\tan\frac{\theta_M}{2})^2 \ . \tag{2.52}$$

Finally, solving for R_{min} yields a quadratic equation which has two roots:

$$R_{min} = \frac{4q_0 \pm 4q_0\left[(1 + 2\tan\frac{\theta_M}{2})^2 + \tan^2\frac{\theta_M}{2}\right]}{2\,(1 - (1 + 2\tan\frac{\theta_M}{2})^2 + \tan^2\frac{\theta_M}{2})} \tag{2.53}$$

The positive solution yields the correct result and R_{max} and W can be obtained via (2.48) and (2.50).

In this way, the sweep width of the radar system can be determined for a given look angle (assuming the center frequency remains constant). If the chirp bandwidth is less than that given above, the range resolution will deteriorate; if it is more, then the extra data are not used.

2.6 Oversampling and Presumming

A typical SAR system samples the demodulated output at a much higher rate than would be necessary, based on the Nyquist rate for the recorded signal. This is because the antenna pattern is not an ideal step function which is limited to the terrain being imaged. Rather, it has

a transition band and sidelobes which can detect targets outside the desired patch. These targets are aliased back into the processed image and result in false targets. By sampling at a higher rate, and then digitally filtering the data prior to interpolation, these outside targets can be suppressed. Sampling in the azimuth direction is done for the same reason, though it is referred to as Doppler oversampling, since the azimuth antenna pattern is obtained through Doppler methods [31]. The oversampled range lines are then low-pass filtered in the azimuth direction to narrow the antenna beamwidth and also to reduce the system noise (coherently summing several range lines into one). It would seem that the prefiltered, highly oversampled data would be better input to the interpolation stage, since the closer spaced data would reduce the interpolation error. This is not done because the storage requirements for such volumes of data are prohibitive. This topic is treated by Brown *et al.* in designing optimal presumming filters which retain azimuthal resolution and allow for limited data storage [32]. Hayner [33] uses the azimuthal oversampling on a keystone grid to analyze the noise effects of a one-dimensional nearest neighbor interpolator and to propose an adaptive presumming filter in the azimuthal direction.

CHAPTER 3

THE INTERPOLATION PROBLEM

Before discussing two-dimensional interpolators, it is useful to examine the one-dimensional case and see what can be extracted from there. Past experience has shown, however, that generalizing from one to multiple dimensions in signal processing algorithms is not as simple as adding a subscript. The problem worsens when the input grid is non-rectangular, and becomes still worse when it is irregular (the sample spacing cannot be parametized). First, we will look at what approximation theory can give us and then some classical interpolation concepts. Local versus global interpolation is discussed, as well as separable versus non-separable interpolation. Then the DSP approach to interpolation will be reviewed. Most classical approximation/interpolation theory texts deal with real data only, although most algorithms can be expanded to the complex domain without loss of functionality.

3.1 Classical Interpolation

Classical *approximation* theory attempts to solve the following problem [34]:

> If V is a normed linear space and W is a subset of V, then given a $v \subset V$, find a $w^* \subset W$ such that
>
> $$\| v - w^* \| \leq \| v - w \|$$
>
> for all $w \subset W$, where $\| \cdot \|$ is a linear norm on the space V

The norm is typically the Chebyshev norm (l_∞) or the Euclidian norm (L_2). In our case, V is the set of two dimensional functions, and W is the set of spatially limited functions, e.g., the ideal low-pass filter (bandlimited) is *approximated* with a truncated or tapered sinc function (spatially limited, but not bandlimited due to the truncation).

Classical *interpolation* theory sets out to solve the following problem:

Given an m partition on the interval I:[a.b] with partition set $X = x_1.x_2.\cdots.x_m$ such that $a=x_1 \leqslant x_2 \leqslant \cdots \leqslant x_m=b$. Let $y_i = f(x_i)$. Find a function $g^*(x)$ such that

$$g^*(x_i) = y_i \qquad 1 \leqslant i \leqslant m$$

and

$$\| f(x) - g^*(x) \| \leqslant \| f(x) - g(x) \| \qquad \text{for} \quad x_1 \leqslant x \leqslant x_N \ .$$

This allows for a very general specification of g(x) which gives rise to the use of piecewise functions in representing g(x), i.e., piecewise polynomials and splines[5]. If the subspace W and V are the same, then an *exact* interpolant can be found, i.e., the function can be reproduced exactly.

It is worth noting a few points about approximation and interpolation for DSP applications. With interpolation theory, we are required to obtain a function which passes through each of the original datum. This has the disadvantage that a noise corrupted signal will be reconstructed with a function passing through each noisy sample, rather than smoothing it out. If the condition that $g^*(t)=y_i$ is relaxed, then we can produce an *approximating* function $g^*(t)$ which minimizes the mean squared error between $g^*(t)$ and the known data points. Let the sample error e_i be defined as

$$e_i = y_i - g(x_i)$$

Find a function $g^*(t)$ which minimizes

$$E_{g(x)} = \sum_{i=1}^{m} | e_i |^2$$

The minimizing function $g^*(t)$ is our interpolant. In this case, however, $g^*(x_i) \neq y_i$ in general, and the original data will not be reproduced by g.

---

[5] Throughout this chapter, the interpolating functions will be written with its argument as t or x interchangeably because x is borrowed from the mathematics area and t from the engineering area.

For approximating a continuous function (signal), we are usually required to approximate it only within a finite interval. This means that our data record is necessarily a fixed length record, i.e., it is time (spatially) limited. If our sequence is of infinite length, as in sample rate changing a voice or communications signal, then it must be processed in blocks. Also, for a fixed length record (such as our SAR problem) and without the use of signal extrapolation, the interpolated reconstruction will be only an approximation to the sampled bandlimited function. The problem of processing a time-limited signal as if it were also bandlimited is ever present. Recall, too, that these interpolation concepts are presented in the time or spatial domain, but are actually applied in the SAR frequency domain where duality is used to justify assumptions about the signal.

3.1.1 One-dimensional DSP interpolators

The following section deals with interpolation from a DSP approach [35]. It demonstrates that interpolation (by a rational factor) is really a filtering operation. The analysis is also carried out assuming that the interpolation is done in the time domain.

Assume that a continuous time function f(t) has finite energy and is σ bandlimited, i.e.,

$$\sum_{t=-\infty}^{\infty} |f(t)|^2 < \infty \tag{3.1a}$$

and

$$F(\omega) = 0 \qquad |\omega| > \sigma \ , \tag{3.1b}$$

where $F(\omega)$ represents the usual Fourier transform of the function f(t):

$$F(\omega) = \int_{-\infty}^{\infty} f(t) \cdot e^{-j\omega t} \, dt \tag{3.2}$$

If f(t) is sampled to produce the infinite sequence x(n) (n, an integer),

$$x(n) = f(nT) \qquad -\infty < n < \infty \tag{3.3}$$

where T is the sample period and $T < \pi/\sigma$, then the function f(t) can be exactly reproduced

with the infinite sum

$$f(t) = \sum_{k=-\infty}^{\infty} x(k) \frac{\sin\left[\frac{\pi}{T}(t-kT)\right]}{\frac{\pi}{T}(t-kT)} \tag{3.4}$$

Equation (3.4) can be recognized as the classical reconstruction theorem for bandlimited signals. It is customary to refer to the interpolation kernel in (3.4) as the function $\mathrm{sinc}(\tau) = \frac{\sin(\tau)}{\tau}$ so that (3.4) becomes

$$f(t) = \sum_{k=-\infty}^{\infty} x(k)\, \mathrm{sinc}\left(\frac{\pi}{T}(t-kT)\right). \tag{3.5}$$

The reconstruction described by (3.5) is equivalent to passing the impulse sequence x(n) through an ideal low-pass filter with cutoff frequency $\omega = \pi/T$. The low-pass filter removes the copies (aliases) of $F(\omega)$ which are replicated every $2\pi/T$ by the sampling operation.

Schafer and Rabiner [36] showed that in the realm of digital signal processing, interpolation is a type of filtering operation. Interpolating by an integer factor L=T/T' is accomplished through first forming a new sequence, v(n), by inserting L-1 zeros between the input data points, x(n) from (3.3), and then filtering this resulting sequence with an ideal low-pass (LP) filter (appropriately scaled).

$$v(n) = \begin{cases} x(n/L) & n = 0, \pm L, \pm 2L, \cdots \\ 0 & \text{otherwise} \end{cases} \tag{3.6}$$

$$y(n) = \sum_{k=-\infty}^{\infty} v(k) \cdot h(n-k) \tag{3.7}$$

where h(n) is the impulse response of an ideal low-pass filter.

The output sequence y(n) will be exact samples of the original function f(t) sampled at T' where T' = T/L. When the input sampling rate T is greater than the Nyquist rate π/σ, the requirements of the LP filter are relaxed so that all that is required is an LP filter in which the

passband is approximately unity for $\pi/T < |\omega|$ and approximately 0 for $|\omega| > \sigma$. Optimal FIR filters can easily be designed which meet these criteria [37], however, there may be an additional requirement that the original sample values remain undisturbed by the filtering operation. This additional constraint may degrade the filter a bit by causing less attenuation in the stopband [35].

If the input sequence is sampled close to the Nyquist rate, the LP filter must then be very close to the ideal in order to avoid aliasing. This usually necessitates a high-order filter which is very costly. This has the same effect as applying (3.5) and then sampling the continuous output f(t) with a sampling period T' (although the continuous function f(t) is of course never really formed in a finite state digital computer). When $T/T' = L/M$ is rational, then we can convert to the new sample rate by first interpolating by a factor of L, low-pass filtering, then decimating by M (retaining only every Mth point).

A difficult problem occurs when T/T' is irrational. In this case, we may never have a point at which x(n) and y(m) coincide, and so the above algorithm cannot be implemented. We must resort to (3.4) for *each* point y(m) (see Fig. 3.1). The infinite sum, however, is impractical, and thus we form a suboptimal interpolation kernel h(t) which has finite support.

$$y(n) = \tilde{f}(nT') = \sum_{k=-K}^{K} x(k)\, h(nT'-kT) \tag{3.8}$$

The notation $\tilde{f}$ indicates that the reconstruction is approximate. In the more general case where y(0) occurs at some t_{offset} such that $y(0) = \tilde{f}(t_{offset})$, then (3.8) becomes

$$y(n) = \tilde{f}(t_{offset}+nT') \tag{3.9a}$$

$$= \sum_{k=-K}^{K} x(k)\, h(t_{offset}+nT'-kT) \tag{3.9b}$$

This suboptimal filter may not remove all of the energy at frequencies above σ, and will consequently generate errors in the continuous reconstruction f(t), and so, in the sequence y(m).

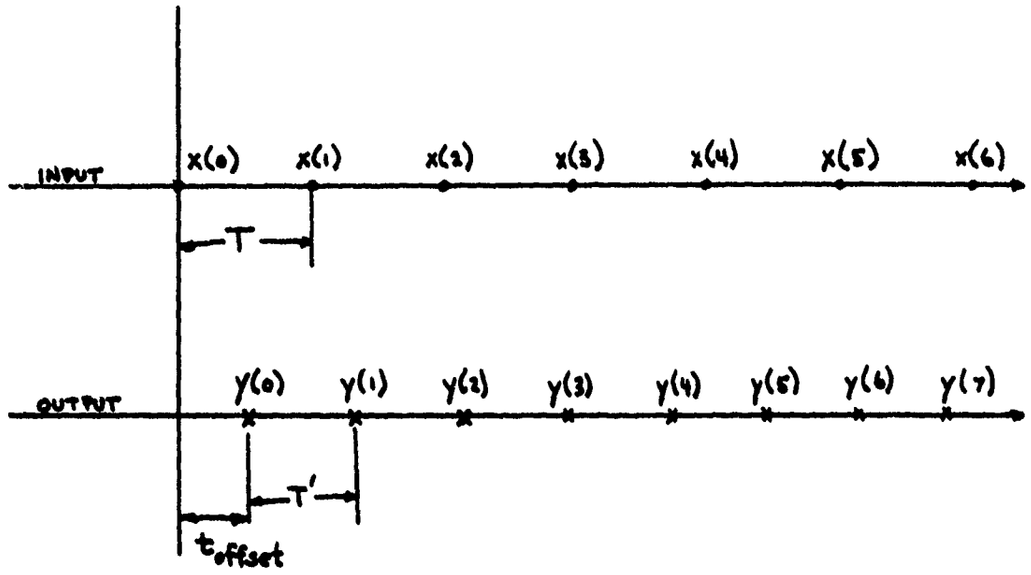


Figure 3.1 Irrational Rate Change with Output Offset.

In the simpler integer ratio rate conversion, $h(\cdot)$ is an FIR filter specified in the digital domain as a sequence. Here, the function $h(t)$ must be a continuous function defined within the limits of the summation, rather than a discrete sequence. This is a consequence of convolving a sampled sequence with a continuous function when the output samples do not match up with the input sample spacings and the rate change is irrational. The continuous time filter must have a transform, $H(\omega)$, which removes the periodic copies of $F(\omega)$ which are above π/T. The performance of the interpolator is determined by how closely the Fourier transform of $h(t)$ approximates the ideal LP filter. That part of the spectrum which is not removed above π/T will not simply show up as high-frequency noise, but will be aliased back into the baseband in the spatial domain when this continuous reconstruction is resampled. This will be discussed more in Chapter 4.

In theory, an arbitrary conversion rate R could be approximated by a rational rate change L/M, but as L and M become large so as to approximate an arbitrary rate change, the filters grow to a very high-order. As an example, a rate change of 1.045 requires L/M = 209/200, that is, interpolating by a factor of 209 and then retaining only every 200th point. At the expense of sample delay, processing can be broken into multiple stages, each of smaller order. Several methods have been designed for breaking down a high-order LP filter into several low-order stages where the decimation is reduced by factors of two [38, 39], or reduced to two stages, each of lowered order[40]. Rabiner and Crochiere present an optimization technique for breaking down the L/M interpolator/decimator (I-D) into a series of K smaller order FIR filters where the optimization is done to minimize the total number of multiplies and adds per second (MADS) [35, 41]. They have shown that by breaking down a high-order, single-stage L/M interpolator/decimator (I-D) into several small order L/M I-D sections, a significant computational savings can be achieved. This notion can be generalized to design more efficient low-pass filters through several interpolation and decimation stages [42]. It should be cautioned, however, that cases can arise in which there is no savings in breaking down the single stage [43].

Ramstad [44] has developed structures for conversion between arbitrary sample rates. He discusses a hardware structure which does real time calculation of the interpolator coefficients, which is essentially a time varying filter. He also presents a "hybrid" interpolator that does rational sampling rate conversion, as previously discussed, followed by a nearest neighbor, linear, or possibly LaGrange interpolator, to generate data which fall between those equi-spaced output samples. The approach is hardware oriented with very specialized structures to generate the time-varying coefficients. While this may be less of a concern for current VSLI technology, it would be useful to have an algorithm that could be implemented in a readily available serial or parallel processor or one of the commercially available DSP chips.

3.2 Global versus Local Interpolators

Nearly all interpolators that are used have local support. That is, the interpolation kernels are finite in length and use input data in only a small neighborhood of the output point. The advantages to this class of interpolators are high speed and low cost. They are faster and easier to implement because only a small, limited number of data points are involved in the summation (convolution). The disadvantage, of course, is a poor signal reconstruction.

From the previous sections, we know that the ideal interpolator impulse response is the sinc function which is of infinite length. This is called a *global* interpolator because it uses sample data (albeit with small weighting for distance samples) from the entire data space to reconstruct the output function everywhere. Of course, for the limited data record such as in SAR, the global interpolator only uses data from the limited set (outside the set, samples are assumed to be zero.) The advantage of the global strategy is the adherence to the fact that a bandlimited signal is analytic (the function must be continuous and continuously differentiable) and thus is determined everywhere by only a piece of the function. This means that when interpolation is performed in the transform domain, *every* point in Fourier space contributes to the spatial domain reconstruction. It is this very reason that low-order interpolators do so poorly in direct Fourier image reconstruction in SAR and tomography.

An example of global interpolation is the zero padding of a DFT sequence prior to performing the inverse FFT to obtain a finer resolution spatial response. This is often used in conjunction with local interpolators to interpolate between rotated, rectangular grids. E.g., to increase image resolution, calculate that image DFT, zero pad by a power of 2, then inverse DFT the zero padded sequence. The resulting spatial domain resolution will be increased by the padding factor. Nearest neighbor interpolation (to be discussed later in this chapter) is then used to obtain samples on the rotated rectangular grid. Speed is gained by means of the FFT, though at the expense of much more memory usage. In a 2-D image of size 64 by 64 pixels (4096 data points), resolution can be doubled (in both x and y directions), but memory is

quadrupled to 16384 data points. For larger data arrays (increased resolution factor), the memory required may be prohibitive. The FFT-zero pad algorithm necessitates that resolution be uniformly increased everywhere so that we cannot simply increase the resolution in some small neighborhood.

It is important to place the zero padding in the correct position of the DFT sequence. The DFT represents uniform samples of the z-transform of the sampled signal where the z-transform of the finite sequence, h(n) is given by

$$H(z) = \sum_{n=0}^{N-1} h(n)\, z^{-n} \tag{3.10}$$

Then, substituting $z = e^{\frac{j2\pi k}{N}}$, we obtain

$$H(k) = \sum_{n=0}^{N-1} h(n)\, e^{\frac{-j2\pi kn}{N}} \tag{3.11}$$

The dc component of H(k) is located at k=0 (z = 1) and the highest frequencies are at k=N/2 (z = -1). The zero pad therefore, must be placed at both ends of the rotated sequence, i.e., in the middle of the sequence H(k). This is at the high-frequency folding point of $H(e^{j\omega})$.

Let the original (time domain) sequence be x(n), for $0 \leqslant n \leqslant N-1$ (assume N even). We wish to increase resolution by a factor of R where R is a power of 2. Assume also that N is even, since most applications which use the FFT fix N to be a power of 2. The DFT of x(n) is

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-j\frac{2\pi}{N}kn} \tag{3.12}$$

The dc point lies at X(0), so we create a new sequence W(k) by splitting X(k) at the high-frequency folding point and inserting L zeroes where L = R·N:

$$W(k) = \begin{cases} X(k) & 0 \leqslant k \leqslant N/2 - 1 \\ 0 & N/2 \leqslant k \leqslant N/2 + L - 1 \\ X(L-k) & N/2 + L \leqslant k \leqslant N+L-1 \end{cases} \tag{3.13}$$

The inverse DFT of W(k) will yield a high resolution version of x(n)

$$w(m) = \sum_{k=0}^{N+L-1} W(k)\, e^{+j\frac{2\pi}{N+L-1}mk} \tag{3.14}$$

This FFT-zero pad algorithm can be very useful for determining the position of a peak in a sampled signal when it falls between DFT bins. It can be used to determine the movement of a point target from one range (or azimuth) cell to another, to calculate the width of the (possibly moved) peak, or to align sampled waveforms which may be out of phase [45]. As an example, consider a sampled periodic sinc waveform (Figs. 3.2 and 3.3). The sinc peak is centered at 4.3, but the sampled sequence, having a resolution of 1.0, could be misinterpreted by assigning the peak position at 4.0. The resolution can be doubled by calculating the DFT of the sequence (Eq. 3.3), zero padding the sequence to 2N, then inverse transforming the result (Eq. 3.4). After this first resolution enhancement, the peak is resolved to 4.5. This can be bettered by further padding the DFT sequence to 4N, 8N, and 16N, each time, doubling the time domain resolution (Figs. 3.4 and 3.5). In one-dimension, the memory usage is linear with the interpolation factor, but quadratic in 2-D. When the zero-pad algorithm is used to interpolate one rectangular grid to another prior to nearest neighbor sampling, the memory required may prohibit more than a simple doubling of the sample frequency.

3.3 Separable versus Non-separable Interpolators

The ideal two-dimensional reconstruction kernel is a bi-sinc, or a separable sinc function with the interpolation kernel

$$h(u,v) = \frac{\sin(\frac{\pi u}{T_u})}{\frac{\pi u}{T_u}} \cdot \frac{\sin(\frac{\pi v}{T_v})}{\frac{\pi v}{T_v}} \tag{3.15}$$

This is easily derived from the one-dimensional case above, and like the one-dimensional interpolator, suffers from the same problem of infinite length which makes it difficult to implement.

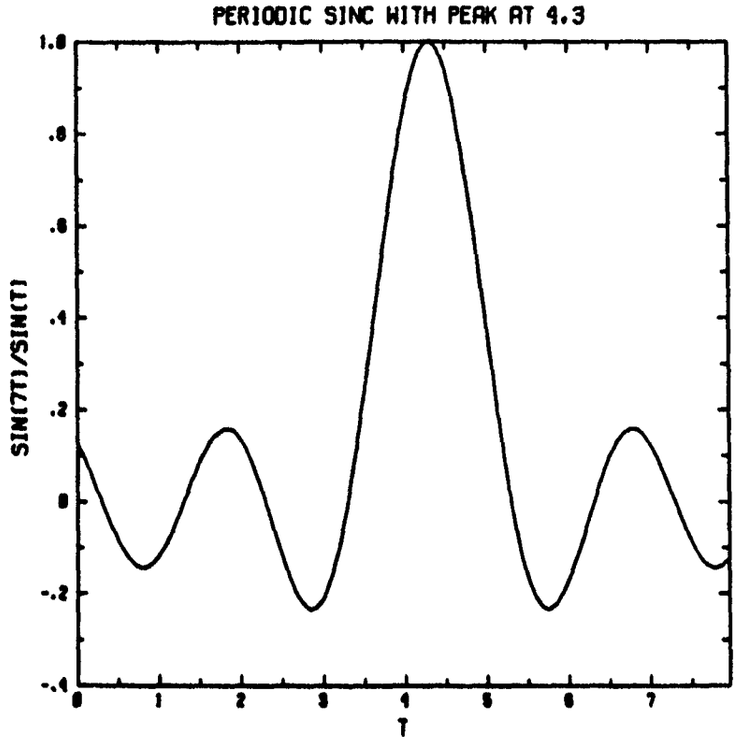


Figure 3.2 Original Continuous Function (Periodic Sinc).

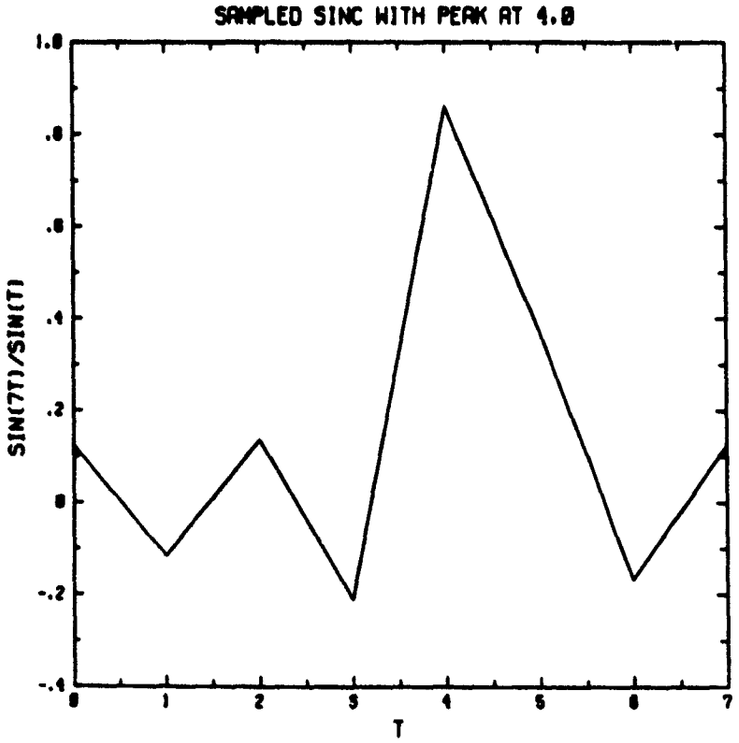


Figure 3.3 Sampled Function in Determining Peak Position.

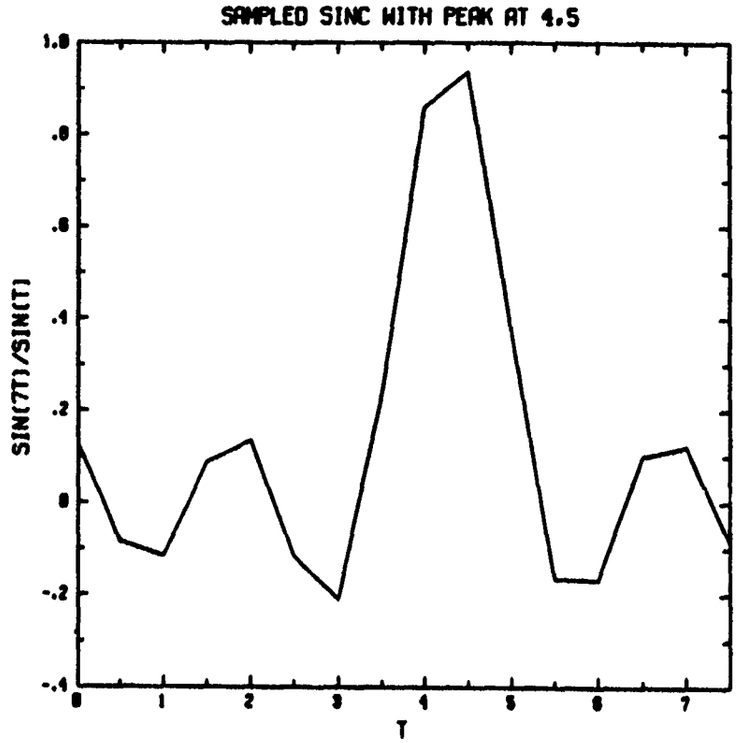


Figure 3.4 Sampled Function With 2 Times Resolution.

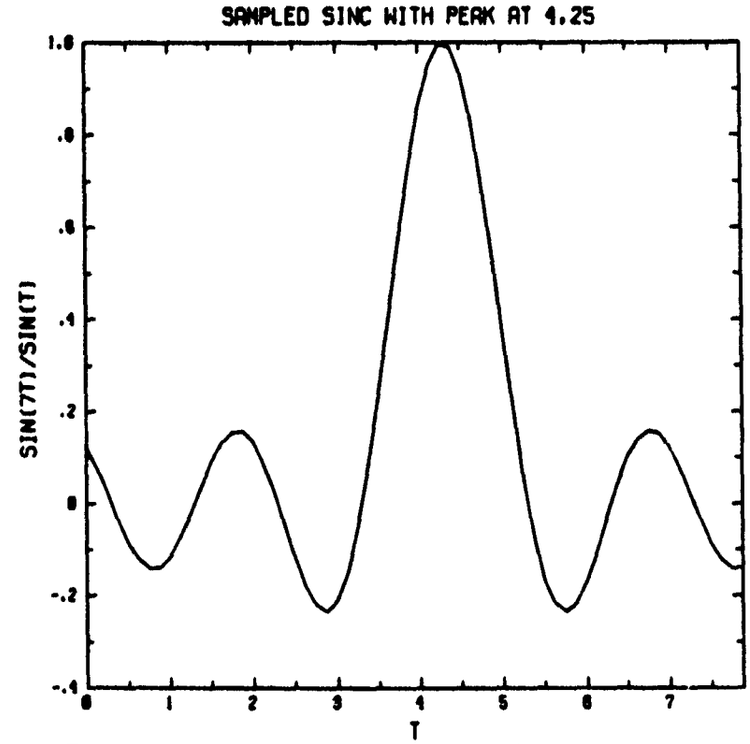


Figure 3.5 Sampled Function With 8 Times Resolution.

More important, the bi-sinc is only separable in U and V, not in R and θ, which is what is required for the polar grid. Thus, what is normally called separable for the Cartesian grid is really non–separable for the polar grid. The separable interpolator is one that can be decomposed into two stages. The first stage interpolates the polar data to an intermediate grid, from which the second stage interpolates to the final rectangular raster. The first stage interpolates the polar data to a *keystone* grid, and the second stage interpolates from the keystone grid to the final rectangular grid. The keystone grid, as indicated by Fig. 3.6, is the intersection of the $\Delta\theta$ spaced range lines and ΔU spaced azimuth lines. When the range lines are equi-spaced in θ, as in a standard polar grid, the azimuthal spacing on the keystone grid is non-uniform. If the pulse repetition frequency (PRF) of the radar is constant, then the azimuthal spacing is constant for each azimuthal line (lines of constant U). The non-uniformity of the azimuth samples for the polar-keystone grids increase the amount of computation required during the second stage of the interpolation.

3.4 A Study of Interpolators

It is clear that the performance of an interpolator is governed by its approximation to the ideal low-pass filter. It is important to have very high rejection in the stopband and be as flat as possible in the passband. Because most of the well-known interpolators are of limited (local) support, they must, by definition, be suboptimal. The following sections describe the spatial domain impulse response and corresponding Fourier transform of several well-known, but rarely characterized, one- and two-dimensional interpolators.

3.5 Nearest Neighbor

The simplest interpolator is the zeroth order, or nearest neighbor. It produces a piecewise constant function (zeroth order polynomial) and is, in general, discontinuous at each breakpoint. The impulse response of the nearest neighbor interpolator is a gate (rectangle) function

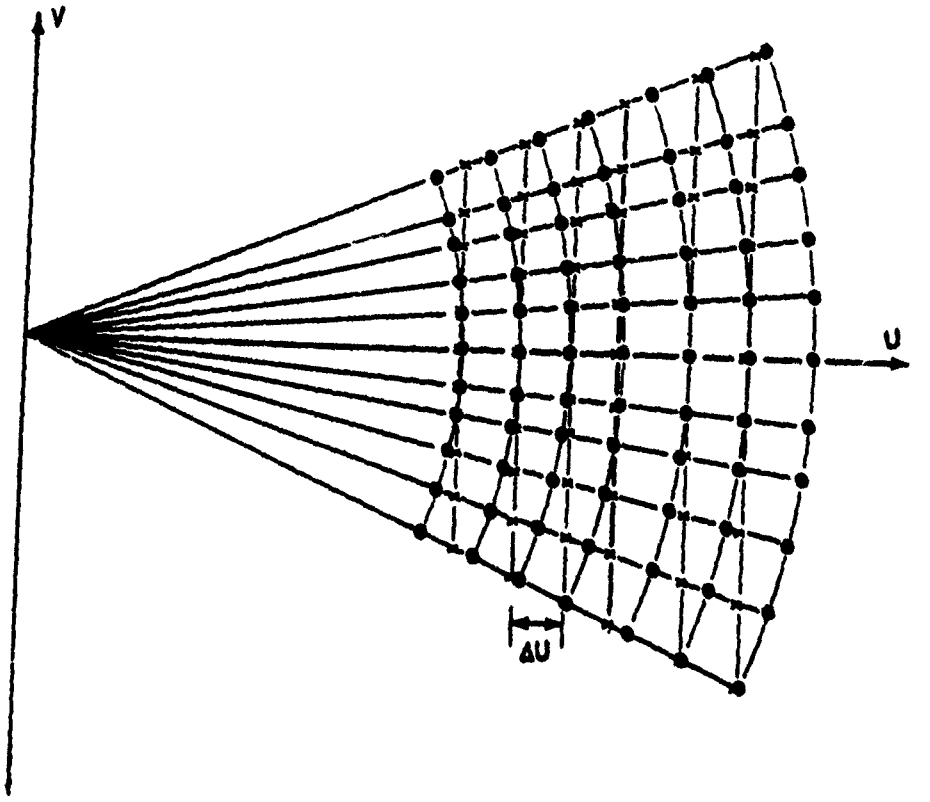


Figure 3.6 Intermediate Keystone Grid.

which is 1 from -T/2 to +T/2 and 0 outside that region (Fig. 3.7).

$$h_{NN}(t) = p_{T/2}(t) \tag{3.16}$$

where

$$p_{T/2}(t) = \begin{cases} 1 & |t| \leq T/2 \\ 0 & |t| > T/2 \end{cases} \tag{3.17}$$

As the gate function is convolved with the input data, exactly one input datum falls within the gate and will be selected for the output. Convolution in the spatial domain implies multiplica-

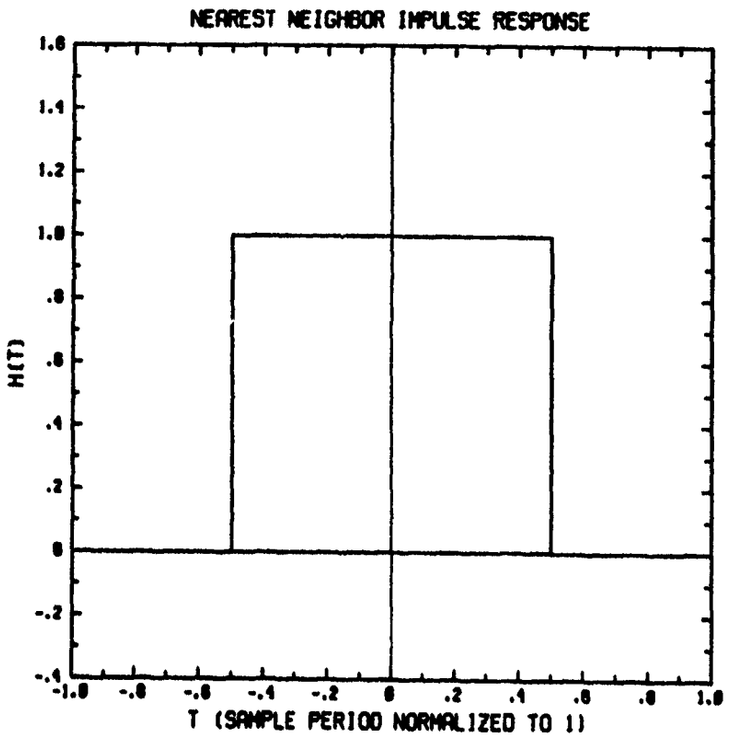


Figure 3.7 Nearest Neighbor Impulse Response.

tion of the transforms in the frequency domain. The transform of this gate function is a sinc waveform with very high sidelobes (Fig. 3.8) and a falloff rate of 3 db per octave.

$$H_{NN}(\omega) = \frac{2\sin(\omega T/2)}{\omega} = T\,\mathrm{sinc}\left|\frac{\omega T}{2}\right| \tag{3.18}$$

It is these high sidelobes which fail to remove the high-frequency copies of the original function. It is, in fact, a very poor low-pass filter.

The nearest neighbor interpolator, as defined above, is also ill-posed for points lying *exactly* half way between known data. The output there would be the sum of the two neighbors. This can be averted by arbitrarily chosing only one of the points. To make the reconstructed function right-continuous, we chose the left point when interpolating exactly at the sample midpoints.

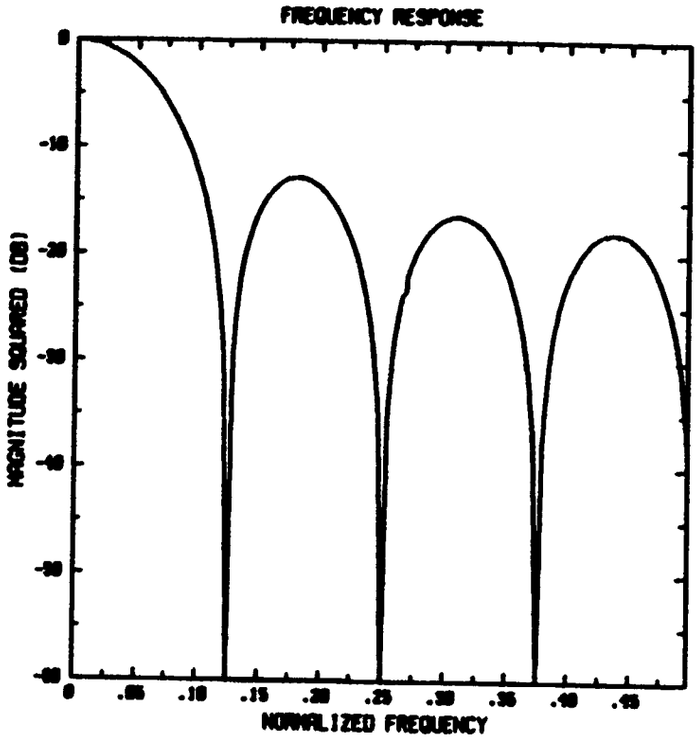


Figure 3.8 Fourier Transform of Nearest Neighbor Interpolator.

In two-dimensions, the nearest neighbor interpolator is a 2-D gate-like function which has a shape that is very dependent on the input data sampling structure. If the input sample spacing is rectangular, then the nearest neighbor function is a spatially invariant rectangular gate pulse (of possibly differing u-v dimensions). If the input array is not uniformly spaced, then it becomes necessary to tessellate the input sample array into nearest neighbor regions (Fig. 3.9). The interpolator then becomes spatially varying and analysis is nearly intractable (in the transform domain) except possibly in a stochastic sense.

On a regular sample grid (e.g., a polar grid) it is possible to analyze this interpolator because it is easier to parametize the 2-D gate. For a polar sample grid, the nearest neighbor function is a spatially varying, rotated trapezoid which grows in size as we move away from the origin, and which rotates with the θ sample lines (Fig. 3.10). The Fourier transform of the

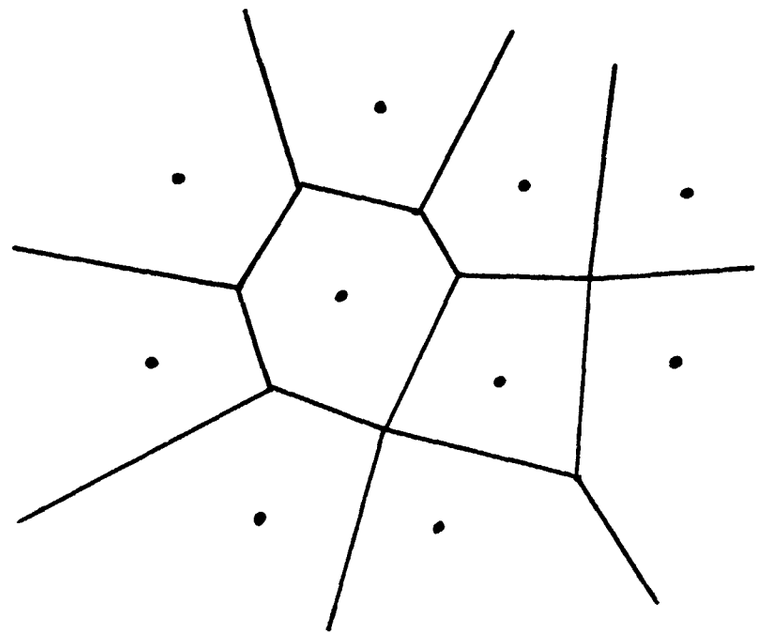


Figure 3.9 Tessellation of an Irregularly Spaced Data Array.

rotated, shifted trapezoid is a bit messy, but straightforward. The spatially varying nature of the transform results in a transform with four rather than two parameters: u,v (frequency coordinates) and r,θ (spatial coordinates). The trapezoid is specified by three parameters: the 2 bases and the width as shown in Fig. 3.11. The Fourier transform of this trapezoid (which has the value 1 inside and 0 outside) can be calculated as follows:

First we see that the trapezoid is really a linearly warped, shifted, and rotated square. The linear warping is described by a simple transformation matrix

$$\begin{vmatrix} u' \\ v' \end{vmatrix} = \begin{vmatrix} \frac{c}{4}(a+b)^2 & 0 \\ \frac{c}{4}(a^2+b^2) & \frac{c^2}{2}(a+b) \end{vmatrix} \begin{vmatrix} u \\ v \end{vmatrix} \tag{3.19}$$

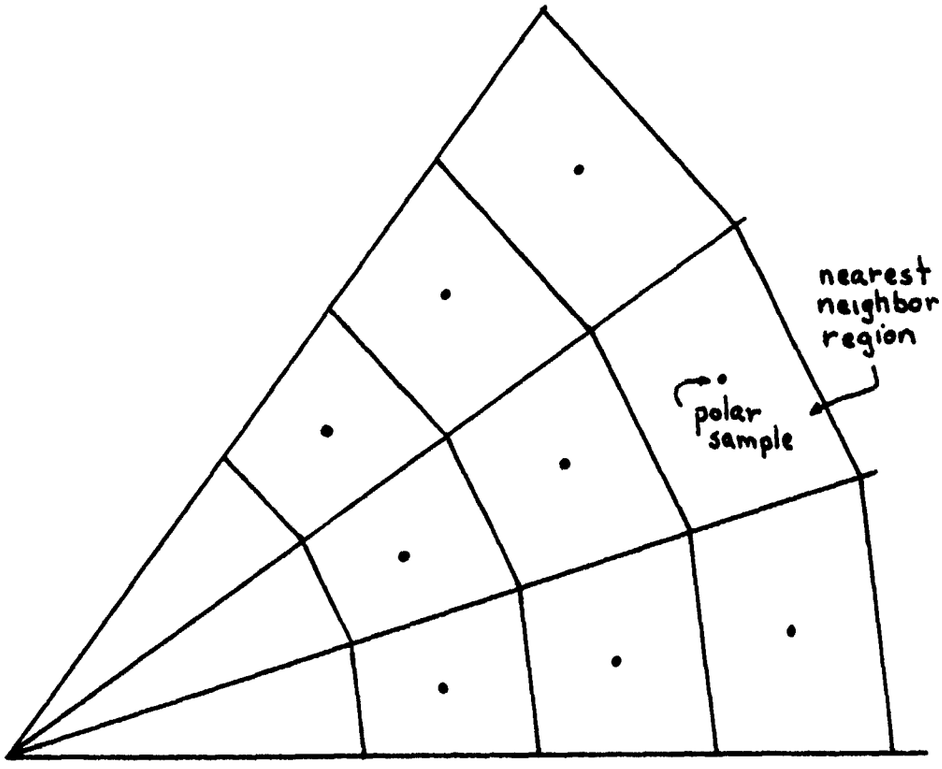


Figure 3.10 Nearest Neighbor Interpolator Geometry for a Polar Grid.

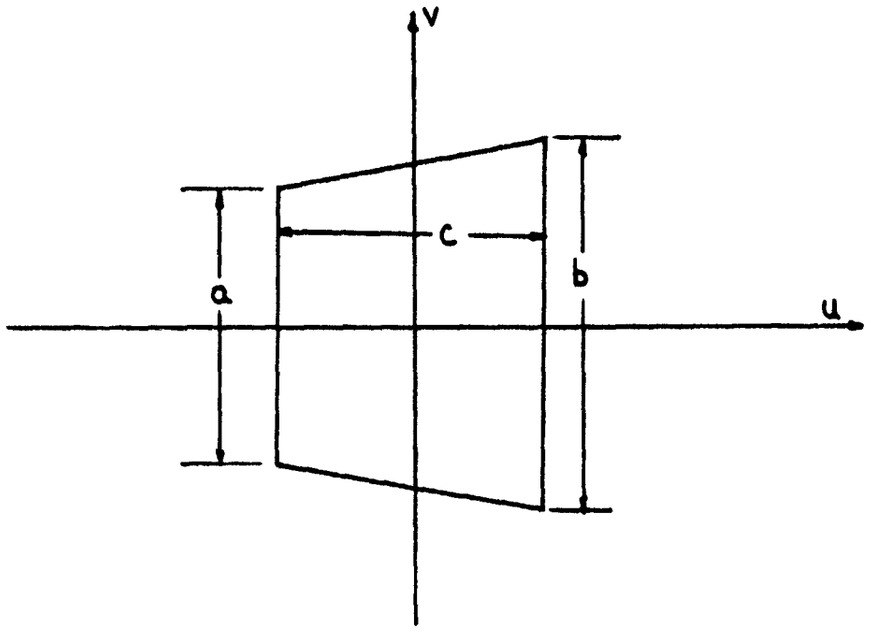


Figure 3.11 Trapezoidal Region for a Warped Sinc Interpolator.

This matrix transforms a square centered at the origin of width one to the trapezoid centered at the origin and with dimension parameters (a,b,c). The trapezoid dimensions (a,b,c) are dependent on the spatial position of the trapezoid. We can now apply a linear shift transformation in the +u direction with u' = u + A, which in the Fourier domain corresponds to

$$\tilde{F}(u,v) = e^{+jAu}F(u,v) \tag{3.20}$$

followed by a rotation. The constant A represents the amount of spatial translation along the U-axis. A is a function of the position

$$\begin{vmatrix} u'' \\ v'' \end{vmatrix} = \begin{vmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} u' \\ v' \end{vmatrix} \tag{3.21}$$

By combining (3.19),(3.20), and (3.21), the spatially varying trapezoidal impulse response and Fourier transform can be determined, though it is extremely awkward to use and therefore this formulation has only nominal value. If we convolve this spatially varying trapezoid with the input polar grid, we end up with a 2-D function that is a piecewise constant surface that looks like blocks of varying heights. This block-like reconstruction is then sampled on the rectangular grid.

A problem that occurs with this form of the nearest neighbor interpolator is the actual implementation. Since it is not easy to determine which trapezoid the rectangular sample falls on, we make a small approximation by bending this shape into a section of an annulus (Fig. 3.12). It then becomes very easy to determine the nearest neighbor with simple integer rounding. The index of the angular coordinate is found by adding 0.5 of the angular sampling spacing to θ, and then truncating to the integer value. The range index is found in a like manner with the range sample spacing. For the particular geometries appropriate in spotlight mode SAR, this is a very close approximation to the ideal. The Fourier transform of the nearest neighbor interpolator is shown in Fig. 3.13.

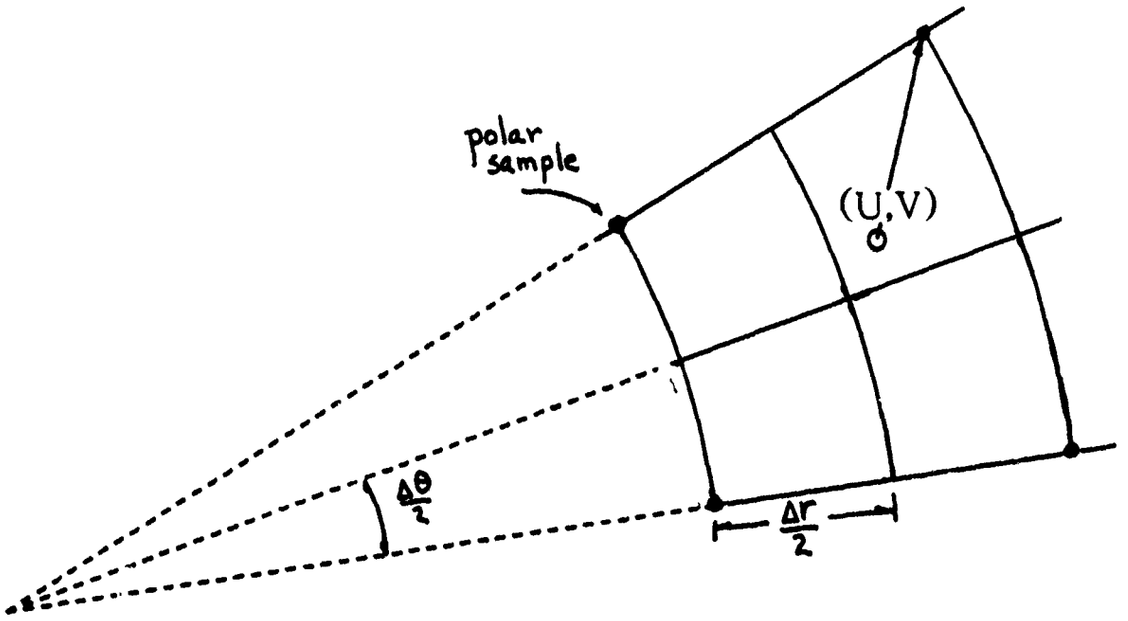


Figure 3.12 Approximation to the True Nearest Neighbor Geometry.

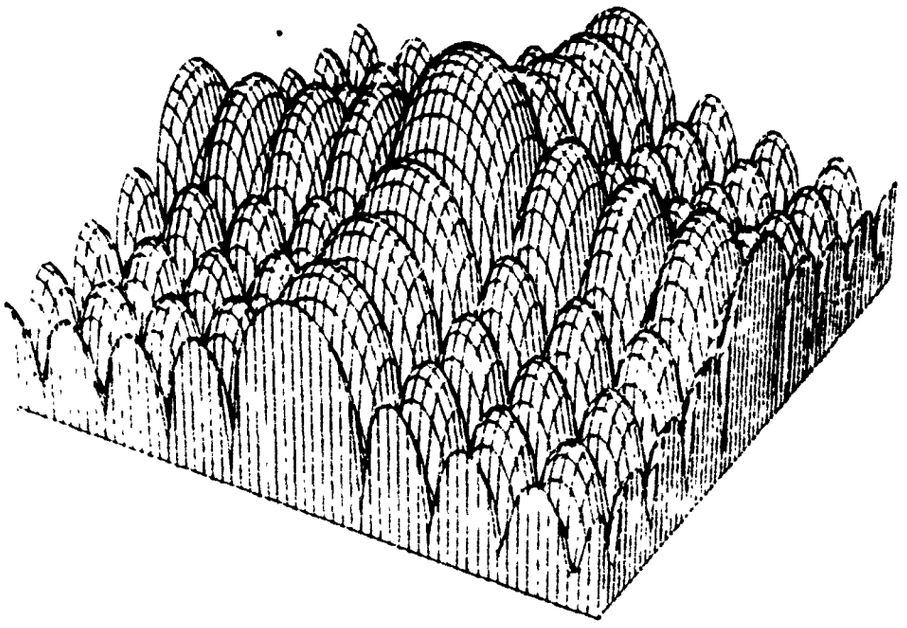


Figure 3.13 Nearest Neighbor Interpolator Fourier Transform.

While the NN algorithm is the crudest of the interpolators presented here, it still requires square-root and arctangent functions to locate the Cartesian point in polar space. The computation, however, can be reduced by expanding these functions in a Taylor series and truncating to 3 terms. The approximation is good enough to locate the point in polar coordinates, and the error in position is usually negligible in effect, i.e., choosing the nearest neighbor incorrectly. In our simulations, the expansion method produced incorrect nearest neighbors in only 3 out of 4096 points - 0.073% error. Of course, the square-root/arctangent functions could also be generated through a lookup table combined with linear interpolation. As we shall see, these small errors in the nearest neighbor coordinate approximations are usually outweighed by the error in the algorithm itself.

When interpolating from one rectangular grid to another, which is merely displaced in U and V (no relative rotation), where the input and output rasters have the same sampling frequency, and are offset from one another by $\Delta U, \Delta V$, the nearest neighbor will merely replicate the original data. In fact, if $|\Delta U| < T_U/2$ and $|\Delta V| < T_V/2$, then the input and output responses will be identical [11].

While this is the fastest and easiest algorithm which can be used to produce Cartesian samples from polar formatted data, it results in a badly distorted response for CAT images [24] that are full of artifacts and false lines. In SAR, it produces many false targets.

3.6 Linear Interpolation

Probably the most widely used simple interpolator is the *linear interpolator*. It is commonly used to "read between the lines" of tables or closely spaced samples. When a function is expanded in a Taylor series about some known point ξ, and then truncated to two terms, the resulting expression is a simple linear curve (straight line) through ξ. A signal which has been reconstructed with a linear interpolator will be continuous, but may lack continuous first and other higher order derivatives which make would it "smooth."

The linear interpolator is usually thought of as a connect-the-dot interpolator. It is more desirable for analytic purposes, however, to find its impulse response. This is the convolution kernel h(t) in our original reconstruction formula

$$y(t) = \sum_{n=-\infty}^{\infty} x(n)\, h\left(\frac{t-nT}{T}\right) \tag{3.22}$$

where T is the *input* sample spacing. One formulation of the linear interpolator is given by (3.23).

$$y(t) = \frac{t}{T}(x_1 - x_0) + x_0 \tag{3.23a}$$

$$= x_0\left(1 - \frac{t}{T}\right) + x_1 \frac{t}{T} \tag{3.23b}$$

It is possible to use input data x_0 and x_1, because the impulse response is shift-invariant for a constant input sample spacing. The subscripts are simply replaced with the points on either side of the continuous variable t.

Comparing (3.23b) with (3.12) gives an h(t) of the form

$$h(t) = \begin{cases} 1 - |t|/T & |t| \leq T \\ 0 & |t| > T \end{cases} \tag{3.24}$$

which is shown in Fig 3.14a. It is important to realize that this is the impulse response which is convolved with the input data to get a linear interpolated (and continuous) output. The transform of this function is

$$H_{linear}(\omega) = \frac{4\sin^2(\omega T/2)}{T\omega^2} \tag{3.25a}$$

$$= T\,\mathrm{sinc}^2\left|\frac{\omega T}{2}\right| \tag{3.25b}$$

which is also referred to as the *Fejer kernel* [46]. It is easy to see that this corresponds to convolving two *nearest neighbor* interpolators, where each is a gate of width T centered at 0 (extending to $\pm T/2$. The convolution of the gate pulses is equivalent to squaring their

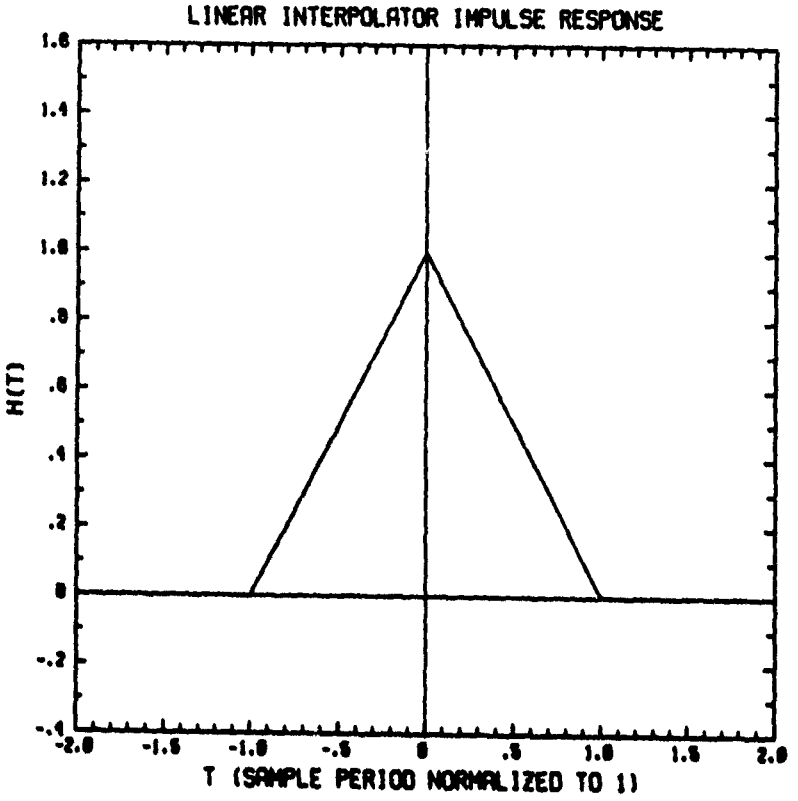


Figure 3.14a Linear Interpolator Impulse Response.

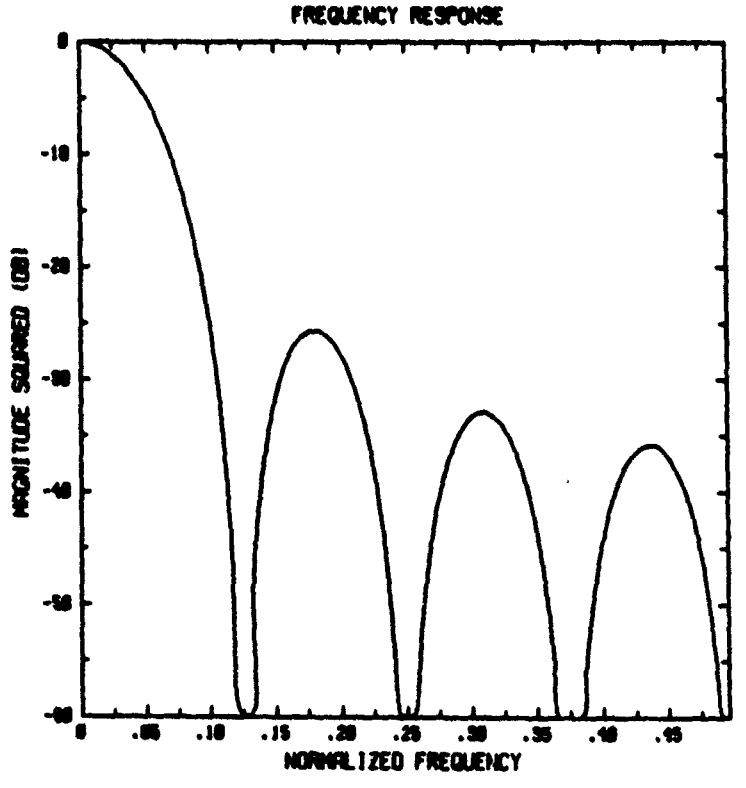


Figure 3.14b Linear Interpolator Fourier Transform.

transforms, resulting in a sinc squared Fourier transform (Fig. 3.14b).

The linear interpolator can also be formulated as an inverse distance interpolator where y(t) is equal to the weighted sum of each of its nearest neighbors. The weights are the normalized reciprocals of their distance to y(t), i.e.,

$$y(t) = \frac{\frac{x_0}{t} + \frac{x_1}{T-t}}{\frac{1}{t} + \frac{1}{T-t}} \tag{3.26a}$$

$$= x_0 (1 - \frac{t}{T}) + x_1 \frac{t}{T} \tag{3.26b}$$

which is identical to (3.23b). It will be shown that this relationship does not hold in two-dimensions.

3.6.1 One-dimensional generalized inverse distance

The linear or inverse-distance algorithm can be generalized to the inverse distance to the *N* (Nth power inverse distance) interpolator. Here, the distances to each of the two nearest data points are raised to some power N, N≥1. This results in weighting the nearer point more heavily than the other. It also results in a continuous first derivative in the reconstructed signal at each interpolated point. The impulse response of the generalized inverse distance (GID-N) interpolator is

$$h(t) = \frac{(1-d)^N}{(1-d)^N + d^N} \tag{3.27}$$

where

$$d = \frac{|t|}{T}$$

This is obviously not a polynomial function, but rather a ratio of polynomials, which are more difficult to analyze. The Fourier transform of a polynomial is easily calculated, but the transform of this class of functions (defined only in the interval [-T,T]) is difficult to

formulate. However, an approximation of the GID-N transform is easily accomplished numerically with the DFT.

In the limit as N approaches infinity, GID-N approaches the nearest neighbor. This is obvious because the weighting of the nearer point approaches infinity (with N) and causes that point to be selected as the output. With very higher order N, T must be scaled to prevent numeric errors due to computer range problems.

3.7 Two-dimensional Generalized Inverse Distance

The linear interpolator can be extended to two-dimensions very simply for rectangular input grids. For this case, the interpolator can be thought of as convolution with a separable 2-D triangle signal (a quadratic shaped pyramid (Fig. 3.15a) shown with equal spacing in u and v) having a half width equal to the u and v input sample spacings. The bilinear interpolator has a Fourier transform which is a separable sinc squared and is shown in Fig. 3.15b. The impulse response for the bilinear interpolator is given by (3.28).

$$h(u,v) = (1 - |u|/T_u)(1 - |v|/T_v) \qquad |u| < T_u, |v| < T_v \tag{3.28}$$

However, when the input grid is non-rectangular, bilinear interpolation is difficult to formulate since the data axes are not orthogonal. The most popular linear type of algorithm used in this situation is the inverse distance interpolator. The output value at Q is the weighted sum of the 4 nearest neighbors (Fig. 3.16).

$$Q = \frac{\sum_{i=1}^{4} \frac{P_i}{d_i}}{\sum_{i=1}^{4} \frac{1}{d_i}} \tag{3.29}$$

The impulse response is given by

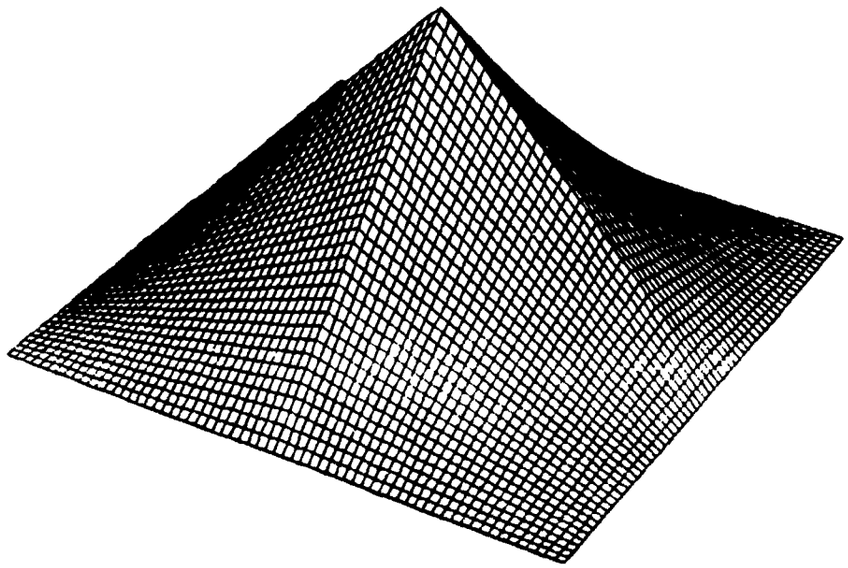


Figure 3.15a Bilinear Interpolator Impulse Response.

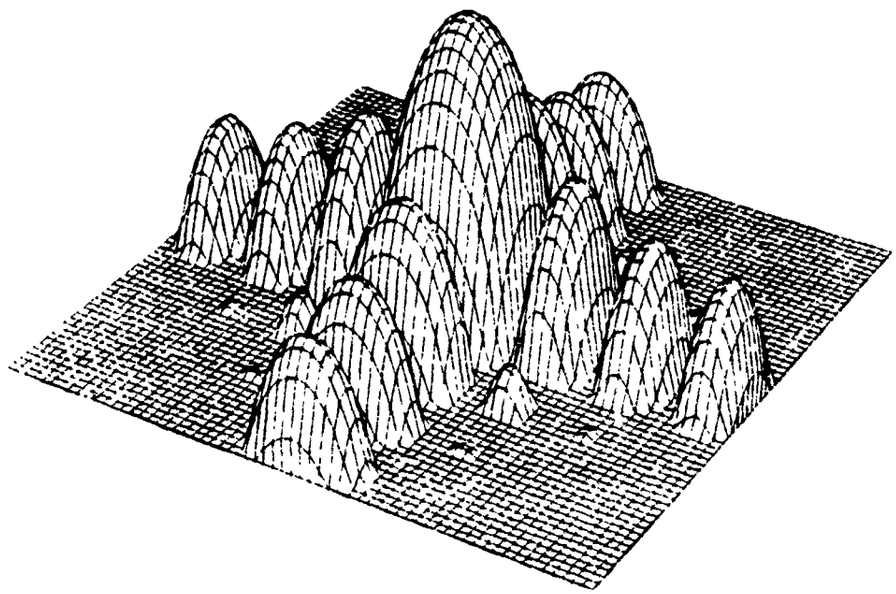


Figure 3.15b Bilinear Interpolator Fourier Transform.

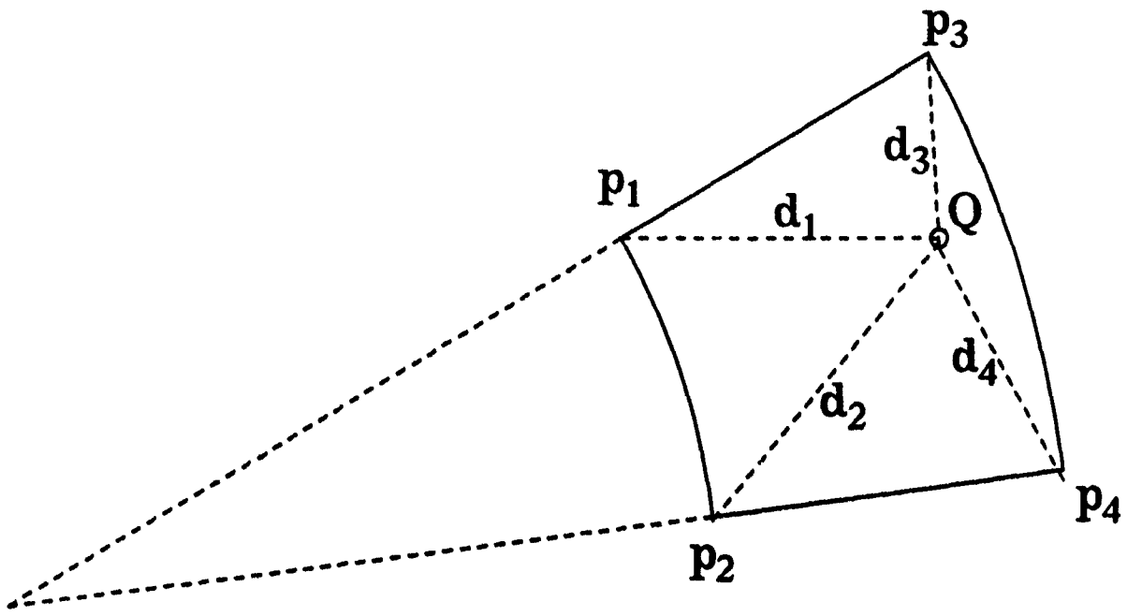


Figure 3.16 Inverse Distance Interpolator Geometry.

$$h(u,v) = \begin{cases} \dfrac{1/d_1}{\sum_{i=1}^{4} 1/d_i} & \max(|u|,|v|) < 1 \\ 0 & \text{otherwise} \end{cases} \tag{3.30}$$

where d_i is the distance from (u,v) to (0,0),(0,±1),(±1,0), and (±1,±1) with the sign depending on the quadrant in which (u,v) falls. If the summation in (3.30) is expanded and the numerator and denominator are each multiplied by $d_1d_2d_3d_4$, h(u,v) becomes

$$h(u,v) = \frac{d_2d_3d_4}{d_2d_3d_4 + d_1d_3d_4 + d_1d_2d_4 + d_1d_2d_3} \tag{3.31}$$

which avoids the problem of singularities when interpolating at an input data point. Notice that h(u,v) has the value 1 at (0,0) and becomes 0 at the eight integer sample points around the

origin (0,0),(0,±1),(±1,0),(±1,±1). This is consistent with the constraint that the interpolator should exactly reproduce the given data at the sample coordinates. The inverse distance impulse response and corresponding Fourier transform are shown in Figs. 3.17a and 3.17b.

The term *Generalized Inverse Distance* actually stems from generalizing the order of the inverse distance weights *and* using additional local points in the summation. These points are determined by extending the point set to include the next *layer* of points around the nearest four. The first additional layer contains twelve more points (sixteen total), the second layer contains eighteen points beyond that (twenty-five total). This admittedly heuristic point set was used by Shepard [47] in contour plotting. It is used here in the computer results to determine the value of the enlarged point set.

It was demonstrated in the 1D case that the linear and inverse distance interpolators were identical. In two-dimensions, this is obviously not the case, since (3.31) cannot be reduced to (3.28). It is non-separable and difficult to analyze, even for the rectangular grid input set. A comparison of the transforms for the bilinear and inverse interpolators, suggest that inverse distance is inferior to the bilinear. It is easy to see (by examining the first derivative behavior of h(u,v) about the (0,0) point) that the output function y(u,v) will not have continuous derivatives. This is similar to the bilinear version, though more pronounced, due to the steeper descent from (0,0). The problem with the first derivatives can be rectified by using inverse distance squared (GID-N, N=2). This choice has a very useful computational side effect.

The inverse distance squared interpolator has some very nice properties. First, the sidelobes are very low (Fig. 3.18b), at the expense of a slightly wider mainlobe.

$$Q = \frac{\sum_{i=1}^{4} \frac{p_i}{d_i^2}}{\sum_{i=1}^{4} d_i^2} \tag{3.32}$$

For the SAR application, this is important because it is important to reduce the sidelobes in order to reduce the artifacts created by the transform operation. Second, the inverse distance

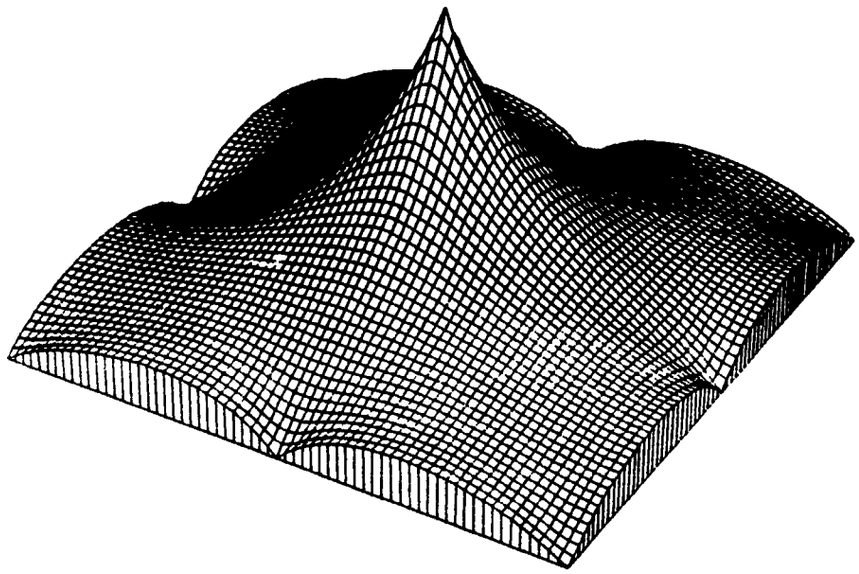


Figure 3.17a Inverse Distance Interpolator Impulse Response.

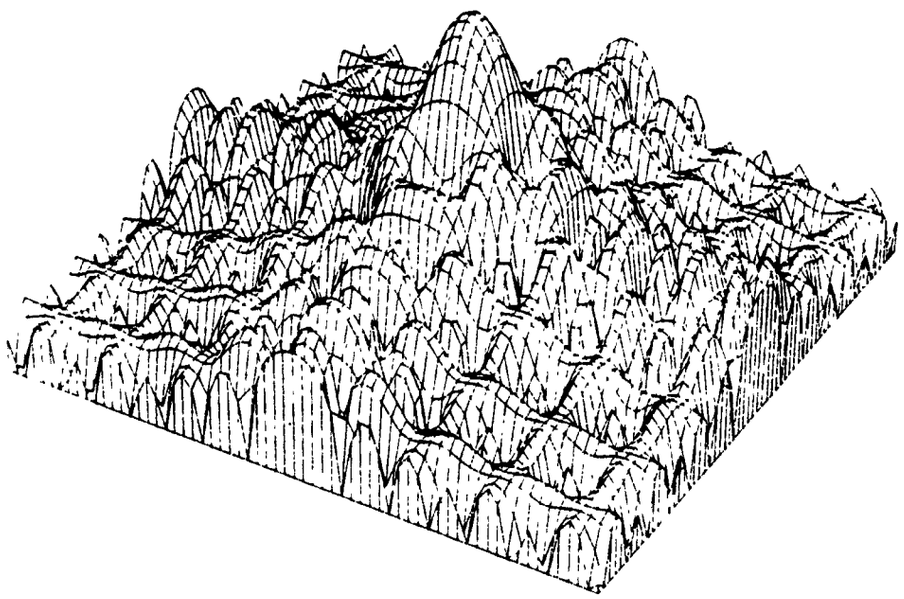


Figure 3.17b Inverse Distance Interpolator Fourier Transform.

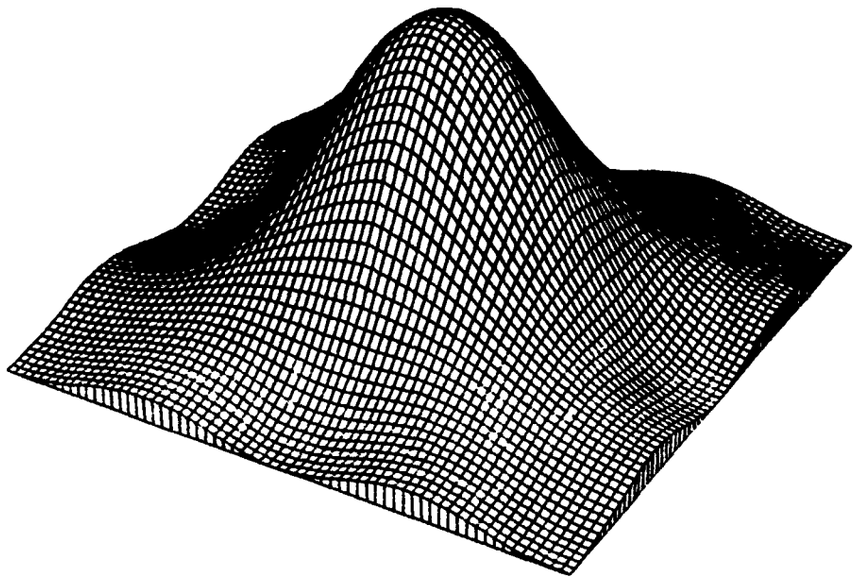


Figure 3.18a Inverse Distance Squared Impulse Response.

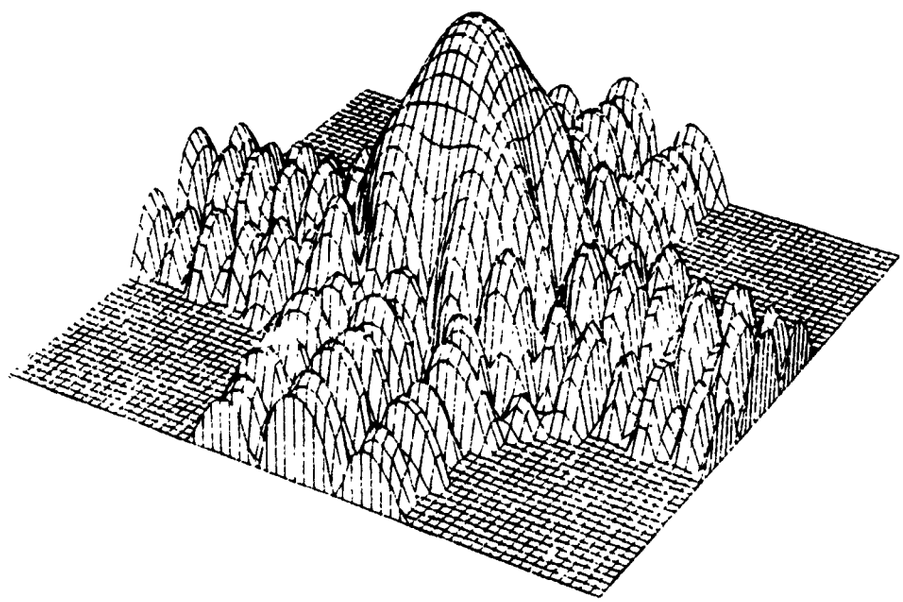


Figure 3.18b Inverse Distance Squared Fourier Transform.

squared algorithm obviates the need to perform the square-root operation in calculating the distances to the nearest 4 points. This is quite a computational savings, as will be shown in the computer simulations. Surprisingly, this algorithm has had little application in the image processing field, but rather has been used more in such areas as contour and 2-D surface plotting (graphics) [47]. Notice that, like the one-dimensional case, as N approaches infinity, the GID-N interpolator approaches the NN interpolator (Figs. 3.19 and 3.20).

3.7.1 Arbitrary local point weighting

In the most general case, when interpolating to a point Q, the nearest P points, p_i, are individually weighted by a function w, a function of the Cartesian distance from Q to P_i, d_i [47, 48]. That is,

$$Q = \frac{\sum_{p_i} W(d_i) \cdot p_i}{\sum_{p_i} W(d_i)} \tag{3.33}$$

The set of points p_i is usually chosen to be the nearest P points, or a set of point within a given circle (used in irregularly space data sets so that areas of sparse data do not used very distance elements in the weighted sum.) w(d) can be any monotonically decreasing function which satisfies (3.34a) and (3.34b).

$$\lim_{d \to \infty} w(d) = 0 \tag{3.34a}$$

$$\lim_{d \to 0} w(d) = \infty \tag{3.34b}$$

This very simple constraint will reproduce the sampled data exactly and can lead to several heuristic weighting functions such as

$$w(d) = \frac{1}{e^d - 1} \tag{3.35}$$

which has an exponential weighting surface. In the previous section, $w(d) = d^n$, where n = 1 for inverse distance weighting and n = 2 for inverse distance squared.

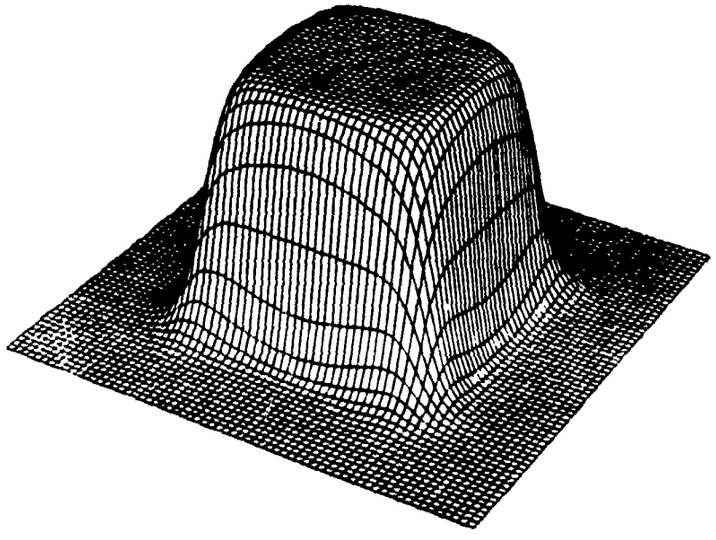


Figure 3.19 Inverse Distance to the 10th Power.

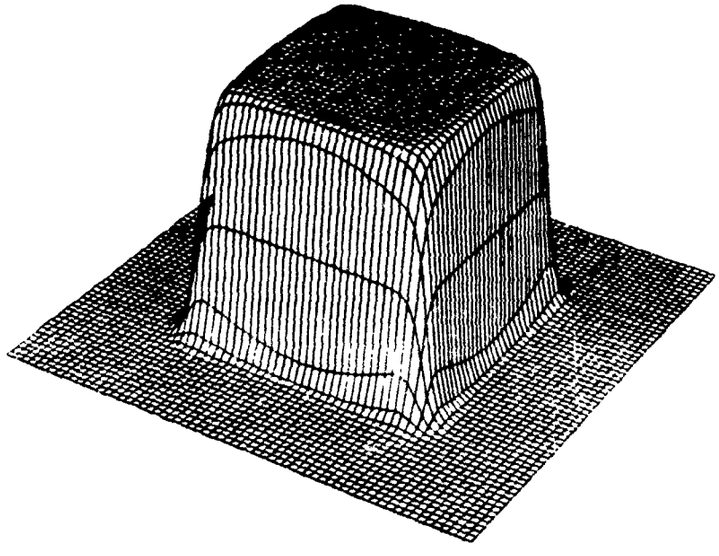


Figure 3.20 Inverse Distance to the 20th Power.

3.7.2 Warped and spatially varying nature of GID

It should be pointed out that the Generalized Inverse Distance interpolator suffers from the same spatially varying problem that the nearest neighbor algorithm does. For a polar grid, the base of the *tent* function (for simple inverse distance) grows larger and more warped. In the Nearest Neighbor section, the frequency response was generated by warping (and rotating) a square gate pulse. This could be done for the inverse distance, but little insight would be gained by such mathematical manipulations. Rather, the very narrow look angle typical of spotlight mode SAR allows us to approximate the warped, spatially varying impulse response with those given above (shown for square input spacing) as a stationary function.

3.8 The Weighted Sinc Interpolator

A third interpolator, henceforth to be called the Weighted Sinc interpolator (also referred to in the literature as the Windowed-Sinc, Standard Polar Format, and ERIM interpolator), w-sinc, is a two-step algorithm. First, the polar samples are interpolated along the range lines to a "keystone raster," that is, a sampling raster which has data on parallel azimuthal lines (Fig. 3.21). Note that this interpolation is implemented using one-dimensional algorithms on a range line by range line basis. In a sense, it is separable, though not in the typical U-V coordinate system, but rather in the R,θ coordinate space. In step two, the data are interpolated, again using one-dimensional techniques, to the final rectangular grid. The computational complexity of this algorithm is determined by the complexity of the 1-D interpolator for each step.

It was shown earlier that the sinc function is the ideal reconstruction filter, although it is impractical due to its infinte support. If we attempt to use a truncated version of the sinc, the resulting transform displays the usual Gibb's phenomenon oscillations which result from high sidelobes and poor passband response. This problem can be alleviated by truncating the sinc function with a tapered window function.

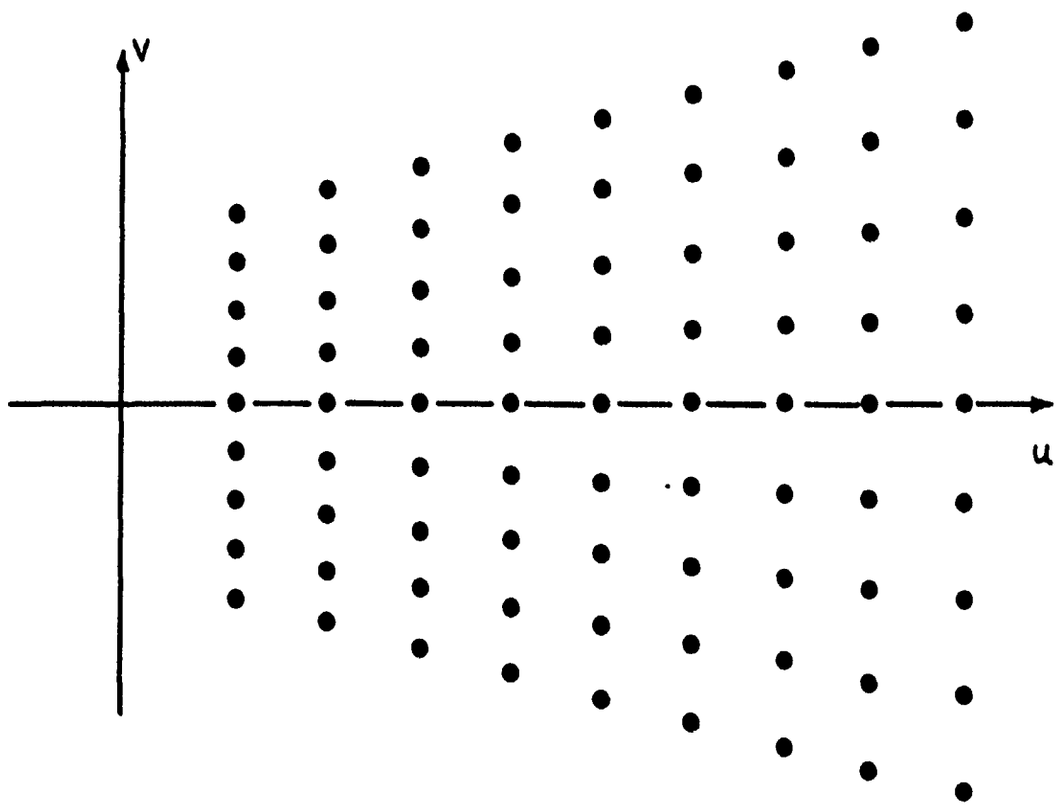


Figure 3.21 Interpolation to an Intermediate Keystone Raster.

$$h(t) = w(t) \cdot \frac{\sin(\frac{\pi t}{T_{out}})}{\frac{\pi t}{T_{out}}} \tag{3.36}$$

where T_{out} is the sampling interval of the output sequence and w(t) is the windowing function. The sinc function is specified in terms of the output sample spacing because it is actually doing the job of low-pass filtering and signal reconstruction in one convolution. The cutoff frequency of the low-pass filter is given by the output spacing so that when we interpolate to a sparser array ($T'>T$), the filtering operation will prevent aliasing of high-frequency data into the low

frequencies. In the first stage of interpolation from the polar grid to a keystone grid, the input sample spacing, ΔR, is constant from range line to range line, but the output sample spacing changes, i.e., the sinc width changes very slightly for each range line interpolation.

Windows are most typically used to taper data records prior to applying a transform (either DFT or IDFT) to reduce the sidelobes caused by truncating the *infinite* sample sequence. Harris [49] has cataloged a large list of well-known windows and their spectrum shaping properties in conjunction with spectral estimation and spectral leakage. Although many windows are candidates for the w-sinc interpolator, the Hamming window was chosen in the computer simulations because of the ease of evaluation and familiarity. The Hamming w-sinc interpolator becomes

$$h(t) = \left| 0.54 + 0.46 \cos\left(\frac{2\pi t}{K}\right) \right| \cdot \frac{\sin\left(\frac{\pi t}{T_{out}}\right)}{\frac{\pi t}{T_{out}}} \tag{3.37}$$

or more simply

$$h(t) = \left| 0.54 + 0.46 \cos\left(\frac{2\pi t}{K}\right) \right| \cdot \mathrm{sinc}\left(\frac{\pi t}{T_{out}}\right) \tag{3.38}$$

where K specifies the support of the sinc calculated in terms of the *output* sample spacing. Because of this specification of K, the number of input samples that fall within $h(\cdot)$ may vary by 1 between any two interpolator sums. This has negligible effects.

The Hamming window is actually a modified Hanning window which is a member of the $\cos^{\alpha}(\theta)$ class ($\alpha = 2$ for Hanning). It can be formulated as the multiplication of a raised cosine by a rectangular window - in the transform domain, it is the convolution of the sinc kernel with three impulses located at 0 (of height 2π), and at $\pm\ \pi/K$ of heights π each. Its transform, therefore, is the weighted sum of three sinc kernels [49]

$$\hat{W}(\omega) = 0.54 \cdot 2\pi\ \mathrm{sinc}\left(\frac{\pi}{K}\omega\right) + 0.46 \cdot \pi\ \mathrm{sinc}\left(\frac{\pi}{K} + \omega\right)) + 0.46 \cdot \pi\ \mathrm{sinc}\left(\frac{\pi}{K} - \omega\right)) \tag{3.39}$$

The side-lobes of the two offset sincs cancel out the sidelobes of the main sinc, and thus, the

overall transform has lower sidelobes at the expense of the wider main lobe.

The Fourier transform of the Hamming windowed sinc is given as the FT of h(t):

$$H(\omega) = FT\{ \operatorname{sinc}(\frac{\pi}{T_{out}} t) \cdot (0.46 + 0.54 \cos(\frac{\pi}{K} t)) \cdot p_K(t) \} \qquad (3.40)$$

which is the convolution of the transforms of each term. The transform of the Hamming window is given in (3.41).

$$H_{Hamming}(\omega) = \frac{(1.08\pi^2 - 0.16K^2\omega^2)\sin(\pi\omega)}{\omega(\pi^2 - K^2\omega^2)} \qquad (3.41)$$

As we increase K, we take more terms into the summation. This has the effect of creating a better low-pass filter, i.e., better interpolator, but also of increasing processing time linearly with K. Figures 3.22 and 3.23 show the impulse response and corresponding Fourier transform of the w-sinc for K=4 and K=10. While it is obvious that the w-sinc shows the most promise for an interpolation kernel, the drawback is the excessive amount of computation required in the lengthy summation (direct convolution), the evaluation of the sine and cosine functions, and the calculation of the input data positions. The data positions of an *a priori* known polar grid could, in theory, be stored in a ROM; however, the data collection path of a maneuvering aircraft precludes such a ROM. Hence, data positions must be calculated "on the fly." Appendix II describes a novel method of reducing sinusoid computation, but the w-sinc interpolator is still costly.

3.9 Splines

3.9.1 Polynomial interpolation

The use of polynomials for function approximation and interpolation is widespread because polynomials are very easy to manipulate, differentiate, and integrate. They are analytic and well-behaved and are representable in a large number of ways. The Taylor expansion of a function is given as a (possibly infinite) polynomial which is often approximated by trun-

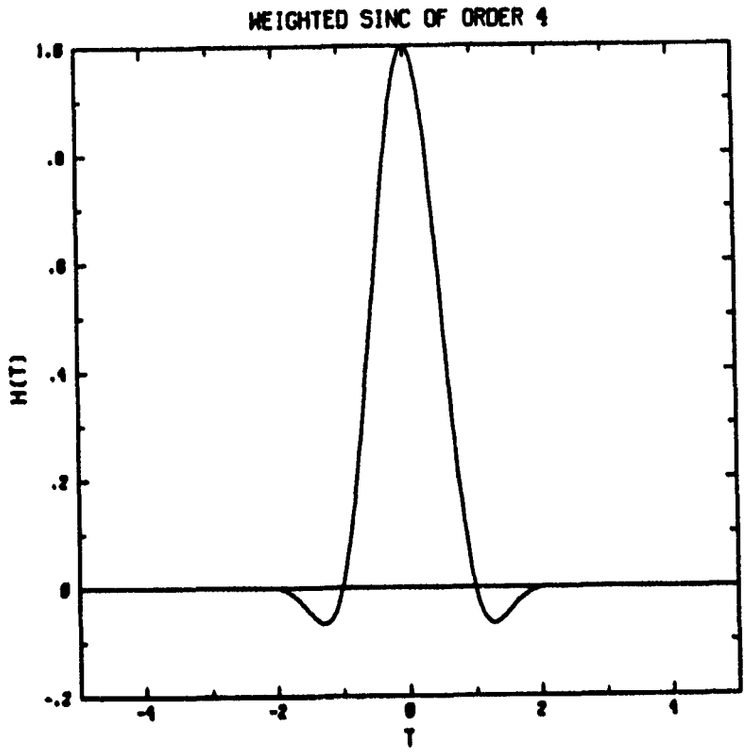


Figure 3.22a Impulse Response of Windowed Sinc of Order 4.

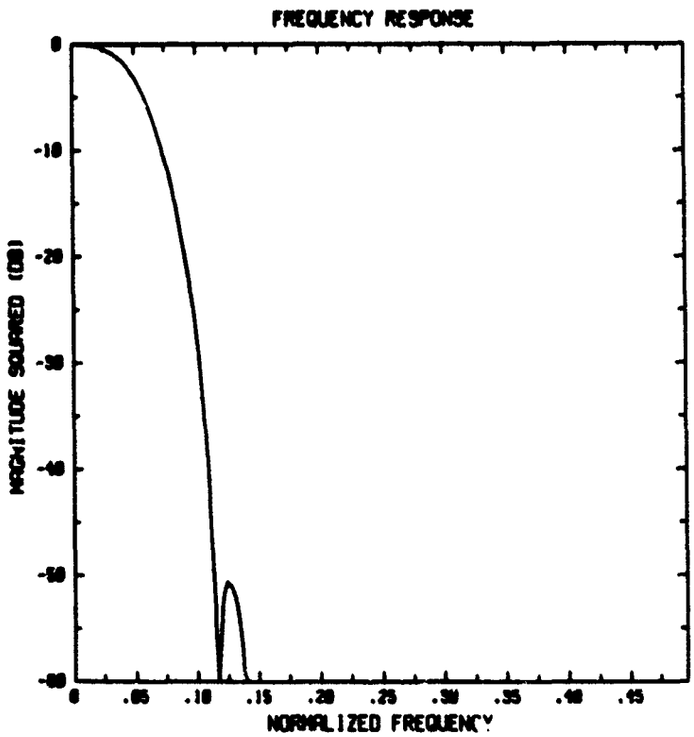


Figure 3.22b Fourier Transform of Windowed Sinc of Order 4.

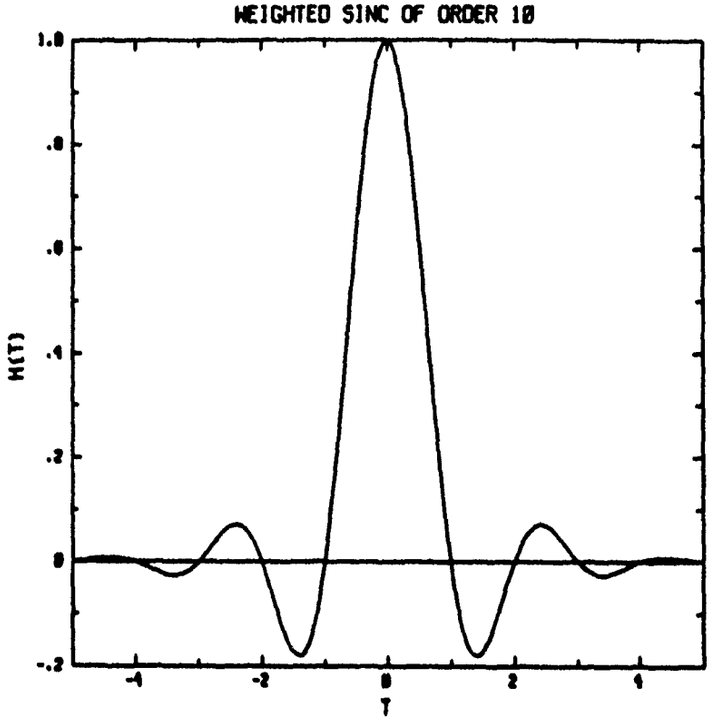


Figure 3.23a Impulse Response of Windowed Sinc of Order 10.

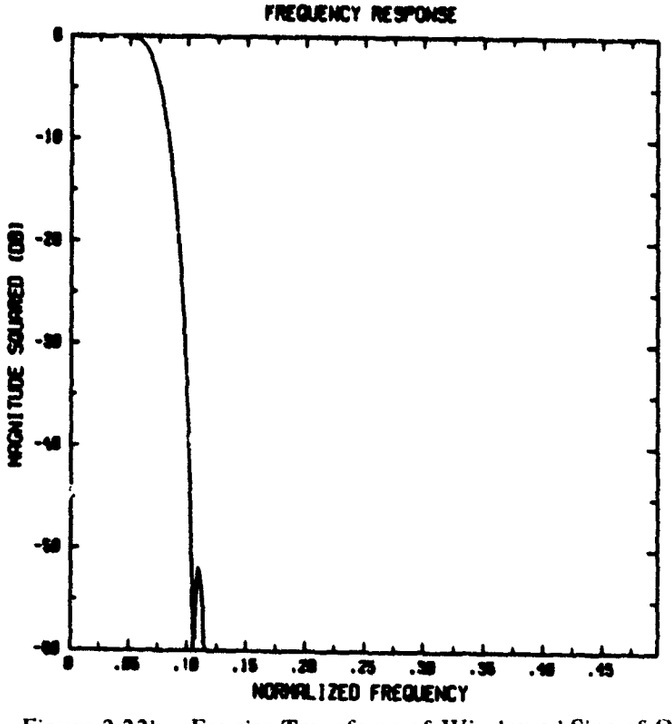


Figure 3.23b Fourier Transform of Windowed Sinc of Order 10.

cating to a finite length polynomial. In computer applications, polynomials are easily represented as an array of coefficients (or derivatives at a specific point). The linear interpolator presented in a prior section is really a first degree polynomial[6]. It is not surprising, therefore, that polynomials could be used for our frequency domain interpolation problem.

Given a set of data points $y_1, y_2, \cdots, y_N$ at time samples $x_1, x_2, \cdots, x_N$, it is well known that an N-1 th degree polynomial, $p_N(x)$, can be generated such that $p_N(x_i) = y_i$. The Nth order polynomial can be written as:

$$p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1} \tag{3.42a}$$

$$= \sum_{i=1}^{n} a_i x^{i-1} \tag{3.42b}$$

where the coefficients a_i can be generated by first forming the intermediate polynomials (called the *LaGrange* polynomials)

$$l_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{N} \frac{x - x_j}{x_i - x_j} \tag{3.43}$$

which have the property that

$$l_i(x_j) = \begin{cases} 0 & i \neq j \\ 1 & i = j \end{cases} \tag{3.44}$$

The nth order polynomial is then generated through the sum of these LaGrange polynomials

$$p(x) = \sum_{i=1}^{N} l_i(x)\, y_i \tag{3.45}$$

Note, however, that this becomes rather unwieldy to work with for large N. It is also computationally unstable on a finite precision computer.

---

[6]Some texts call it a *second order* polynomial because it is specified by two coefficients - it spans a two-dimensional space. The terms *degree* and *order* shall be treated here as the highest power of the polynomial, i.e., a cubic is a third degree polynomial.

Another problem with polynomials of increasing degree is convergence. As the number of interpolating points is increased, the interpolating polynomial $p_N(x)$ is not guaranteed to converge to the original function $f(x)$. In fact, it can be shown that for any sequence of sets of points x_i, there is a function $f(x)$, such that the sequence of interpolating polynomials will diverge at these points [34]. Finally, the multidimensional polynomial is not unique for a given set of points, i.e., a rectangular grid of sample points will not produce a unique polynomial of the form

$$p(x,y) = a_n x^n + a_{n-1} x^{n-1}y + a_{n-2} x^{n-2}y^2 + \cdots + a_1 y^n + a_0 \ . \tag{3.46}$$

This means that it cannot even be used as a local 2-D interpolator, precluding its use for the SAR system.

Though there are various ways to represent a polynomial which can reduce the amount of instability in computation (Newton's form with divided differences, for instance), these Nth order functions tend to oscillate too much between the nodes. This, of course, is because of the fact that an Nth order polynomial must have N roots. On a digital computer, the high-order terms of the polynomial are very sensitive to small changes in x. This is sometimes referred to as an ill-conditioned function. One well known technique to deal with these shortcomings is by using different lower order polynomials in different regions of the curve. The result is the class of polynomials called *piecewise polynomials* or pp functions. Another name for pp functions is *splines*.

The term *spline* has a rather broad definition that is frequently misused in the literature. Most simply, a spline is a piecewise polynomial without constraints, i.e., nothing is said about continuity of the spline or continuity of any of its derivatives. Typically, however, the spline is constrained to be at least continuous, and m derivatives of the Nth order spline are to be continuous, where $m \leq N-1$. The linear interpolator can be referred to as a first-degree spline which is continuous, but lacks any continuous derivatives. A parabolic spline is a piecewise quadratic function which has a continuous first derivative. A cubic spline is continuous and has

continuous first and second derivatives everywhere on the interval.

Spline functions are often used in mathematical and curve-fitting applications, because they can generate smooth interpolating functions with continuous derivatives at the known data points (also called "knots," "joints," and "breakpoints") [50, 51, 52]. More recently, splines have been used in computer graphics to generate smooth curves from discrete edge points of objects, and thus reduce the number of points required for an irregular curve.

Splines have also been used in typesetting applications where enlargement and reduction of character sets are more easily accomplished with pen (plotter, laser, light) strokes than with binary digitizations [53]. By storing character sets as a combination of straight lines and spline curves, they can be arbitrarily scaled without degradation.

3.9.2 Cubic spline interpolation

To see how splines can be constructed to be smooth, i.e., to possess first and second derivatives at each knot, we construct a simple case with only two knots with known values and derivatives. The following is taken from Rivlin [34].

If $\alpha < \beta$, and

$$\begin{aligned} p(\alpha) &= u_1 \qquad p(\beta) = u_2 \\ p'(\alpha) &= u'_1 \qquad p'(\beta) = u'_2 \end{aligned} \tag{3.47}$$

then

$$\begin{aligned} p(x) = u_1 &\left[\frac{(x-\beta)^2}{(\beta-\alpha)^2} + 2\frac{(x-\alpha)(x-\beta)^2}{(\beta-\alpha)^3} \right] \\ + u_2 &\left[\frac{(x-\alpha)^2}{(\beta-\alpha)^2} - \frac{2(x-\beta)(x-\alpha)^2}{(\beta-\alpha)^3} \right] \\ + u'_1 &\frac{(x-\alpha)(x-\beta)^2}{(\beta-\alpha)^2} + u'_2 \frac{(x-\alpha)^2(x-\beta)}{(\beta-\alpha)^2} \end{aligned} \tag{3.48}$$

Because a cubic polynomial has 4 degrees of freedom, the polynomial can be forced to have a

given value and derivative at each of 2 points.

If the spline is constrained to have a continuous second derivative, then it can be shown that the entire spline, $s(x)$, which consists of the cubic polynomial sections, $s_i(x)$ for $x_i \leq x \leq x_{i+1}$, can be uniquely specified by the given breakpoints x_i, the sample values y_i, and the boundary conditions u'_1 and u'_N. The system of equations thus depends on the initial and final derivative of the knot sequence. The end conditions can also be specified as second derivatives, or constraints on the first and/or second derivatives which simply increase the system of equations which must be solved (by 2 more equations). It is interesting to note that spline interpolation has a global nature. Each datum in the set of N points affects the entire spline function, though in a much more subtle way than the Nth order polynomial interpolant. DeBoor showed that this system of equations forms a band symmetric matrix which can be inverted by Gaussian elimination *without* pivoting [54]. The recursive solution is very fast and is implemented in the IMSL mathematics programming library.

The various boundary conditions which can be imposed on the spline function are discussed here. Unfortunately, the names of the more popular ones are not wholly descriptive of their effects. If the derivative of the original function is known at the end points, then we select $u'_1(x_1) = f'(x_1)$ and $u'_N(x_1) = f'(x_N)$. The resultant spline is called the <u>complete</u> cubic spline of f. Most DSP applications do not have derivative values available, nor are they estimated well due to additive system noise. If the second derivatives at the endpoints are forced to be 0, then $s(x)$ becomes the <u>natural</u> spline. This is a popular boundary condition in available software, but is rather arbitrary and leads to only $O(|t|^2)$ convergence, which means that as we decrease the spacing between the knots, the function $s(x)$ converges to $f(x)$ in a squared fashion. By contrast, the complete spline interpolant converges in $O(|t|^4)$ to $f(x)$.

Another possible boundary condition is called <u>not-a-knot</u>. In this case, the first and last knots of the sequence, x_1 and x_N, become inactive and $s_1(x) = s_2(x)$ and $s_{N-1}(x) = s_{N-2}(x)$. This, too, has $O(|t|^4)$ convergence to $f(x)$. It is the end condition used in the IMSL math library for

spline interpolation.

The cubic spline function is given by piecewise cubic polynomials of the following form [54]:

$$s_i(x) = c_{1,i} + c_{2,i}(x - \xi_i) + c_{3,i}(x - \xi_i)^2 + c_{4,i}(x - \xi_i)^3 \qquad x_i \leqslant \xi_i < x_{i+1} \tag{3.49}$$

where the coefficients $c_{i,j}$ are chosen such that

$$s_i(x_i) = s_{i-1}(x_i) \qquad 1 \leqslant i \leqslant N \tag{3.50a}$$

$$s'_i(x_i) = s'_{i-1}(x_i) \qquad 2 \leqslant i \leqslant N-1 \tag{3.50b}$$

The constant coefficient is set to $c_{1,i} = y_i$ so that the spline interpolates to the input data set at the knot positions. Once these coefficients have been determined through the Gaussian elimination step, it is a simple matter to locate the interpolation point within the correct interval and compute the cubic polynomial. If many points are to be generated within an interval, this method would be more preferable than the weighted sinc from a computational view.

The cubic spline interpolant has the unique property of maximal smoothness. That is, for a given set of interpolation points, the cubic spline gives the smoothest curve over all functions. The smoothness is defined as the integral of the square of the second derivative:

$$\int_a^b [f''(t)]^2 dt \tag{3.51}$$

This integral is minimized over all f(t) by s(t), the spline interpolant. It is also termed the *roughness* [55]. The smoothness integral is an approximation to the strain energy function

$$\int_a^b \frac{[f''(t)]^2 dt}{[1 + (f'(t))^2]^{5/2}} \tag{3.52}$$

This is, in fact, where the term spline is derived. A spline was a thin rod used by draftsman to fair curves through a set of data points. The curve naturally minimized its strain energy and is approximated by the cubic spline function.

Because the spline interpolating function is global and requires a matrix solution, it does not possess a time invariant impulse response. Horowitz has determined the power spectral density effects of spline reconstruction [56]. He found that higher order splines are needed to preserve the power spectrum of the original function, but that these higher order polynomials require much more computation to generate and evaluate. A close cousin to the complete spline interpolant is formed via the B-spline functions.

3.9.3 B-splines

Hou and Andrews [57] used cubic basis splines, or B-splines (sometimes called β-splines), for image enlargement and reduction, and for text magnification and minification, with the conclusion that the B-spline may prove more appropriate than a truncated sinc for finite length data records.

The term B-spline comes from the basis formulation of the reconstruction function. If $f(t)$ is the original sampled function (to be reconstructed) at the points $x(nT)$, then we construct a set of basis functions, $B_i(t)$, and form the sum

$$\hat{f}(t) = \sum_{i=1}^{K} c_i B_i(t) \tag{3.53}$$

Note that the c_i are not the original samples y_i, but rather new coefficients calculated from y_i. For equally spaced data, the B-spline functions are shift invariant and the subscript i can be dropped from B.

The cubic B-spline is formally defined as the convolution of two triangle (Chateau functions) pulses, each of width T and height 1. The resulting function is given as two cubic polynomials and is symmetric about the origin. The interpolation kernel is given by:

$$h(t) = \begin{cases} t^3/2 - t^2 + 2/3 & |t| < 1 \\ -t^3/6 + t^2 - 2t + 4/3 & 1 \leq |t| < 2 \\ 0 & |t| > 2 \end{cases} \tag{3.54}$$

This can be recognized as the Parzen window which has a Fourier transform (scaled so that $F(0) = 1$).

$$\hat{F}(\omega) = \left| \text{sinc}\left(\frac{\omega T}{2}\right) \right|^4 \tag{3.55}$$

This is expected since the convolution of two triangle waveforms corresponds to the product of its transform, which are in turn resulting from the convolution of two gate pulses. The nth order B-spline can be recursively generated by convolving the (n-1)th order B-spline with a gate pulse of width T.

This formulation of spline reconstruction is actually a different interpretation of the complete spline given above. The implementation is slightly different, but the dependency of c_i on y_i is global, and thus while the B-spline does have finite support, the B-spline coefficients do not. The interpolation function is generated by computing the c_i's and then filtering this sequence with the B-spline kernel. The previous interpretation leads to a faster implementation and is used in the computer analysis. A B-spline type of reconstruction can be done with a slight modification to the basis function, which leads to an interpolation impulse response that resembles a truncated sinc function.

3.9.4 Cubic spline convolution

The complete spline interpolant is generated by solving a system of equations which come about by the constraints on the interpolating points, the first and second derivative matching at these points, and the boundary conditions. A more general cubic spline that does not have all these constraints is given by the following piecewise cubic polynomial which is symmetric about the origin [11]:

$$h(t) = \begin{cases} a_3t^3 + a_2t^2 + a_1t + a_0 & |t| < 1 \\ b_3t^3 + b_2t^2 + b_1t + b_0 & 1 \leq |t| \leq 2 \end{cases} \tag{3.56}$$

To create an impulse response that interpolates the data exactly, f(x) must be 0 at t=1 and t=2. Also, first derivative of each cubic should match at t=1. This gives seven constraints in all, but the piecewise function has eight constants. This constant can be made a variable in the function f(x) which can be modified to generate different cubic spline functions. It is important to remember that this spline is different from the complete cubic spline interpolant presented previously.

$$h(t) = \begin{cases} (c+2)t^3 - (c+3)t^2 + 1 & |t| < 1 \\ ct^3 - 5ct^2 + 8ct - 4c & 1 \leq |t| \leq 2 \end{cases} \tag{3.57}$$

When the constant c has the value -0.5, the interpolation error goes to zero with the third power of the sample interval ($O|T^3|$). The second derivatives of the two polynomials are forced to be equal if c = -0.75. This same result was used by Keys [58] who demonstrated that the accuracy of the interpolation is between the linear and complete spline interpolant. This form of cubic spline convolution is studied in the computer analysis of Chapter 5. The impulse response of the modified B-spline with c = -0.5 is plotted against the w-sinc of order 4 in Fig. 3.24a. Note that this B-spline is remarkably similar to the w-sinc. Similar performance is expected. The Fourier transform of the B-spline (c = -0.5) is presented in Fig. 3.24b.

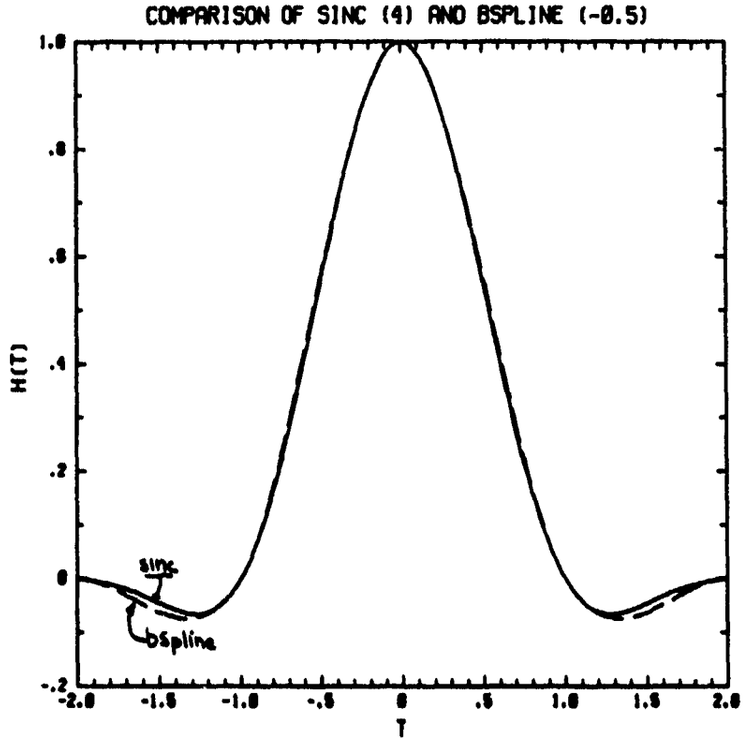


Figure 3.24a B-spline (c = -0.5) Impulse Response with Sinc (order = 4).

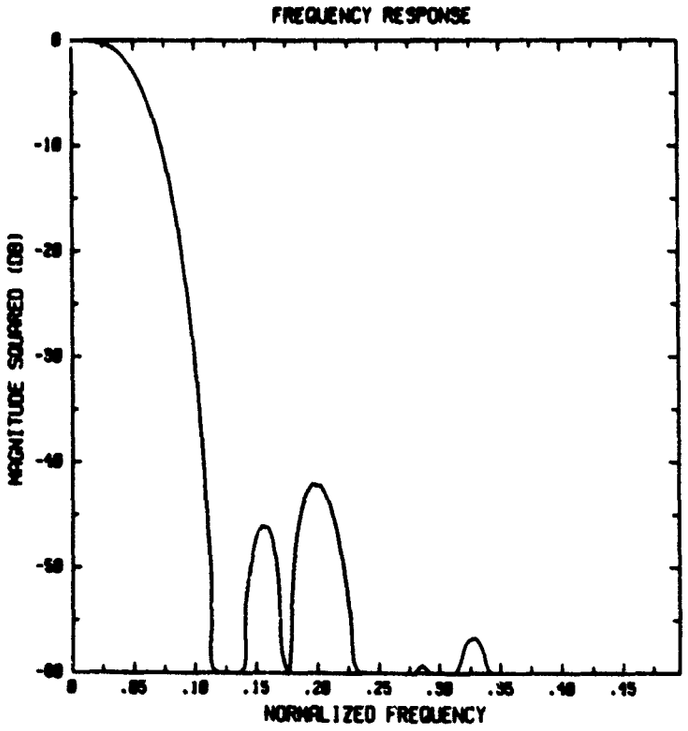


Figure 3.24b Fourier Transform of B-spline (c = -0.5).

CHAPTER 4

WINDOWS, SPURIOUS TARGETS AND ALTERNATIVE SAMPLING GRIDS

This chapter presents a number of topics that are important in SAR processing. Window design for SAR data arrays is discussed, and the usefulness of optimal windows is examined. Spurious target generation resulting from the Fourier domain interpolation stage is presented, and some one-dimensional examples are shown. Finally, alternate sampling grids are proposed which can reduce the computation and even the complexity of the sampling hardware.

4.1 Windowing the Data Array

It has been demonstrated that spotlight mode SAR can generate a data set which is approximately the Fourier transform of the complex reflectivity map. This is inverted with the DFT (FFT). The same procedure is used for the tomographic reconstruction problem as well as remote sensing arrays (ground-based aeronomy SARs). However, when using the DFT as an approximation to the Fourier integral, errors occur due to the limited size and particular shape of the recorded data. It is well-known that strict truncation (uniform windowing) of a signal results in Gibb's oscillations, or sidelobes, which can obscure weak signals. These sidelobes can be reduced through the use of a weighting function, or window which tapers to zero near the ends of the sampled data set. In addition, this tapering smooths out discontinuities at the end points (the DFT assumes that the data represents one period of a periodic signal, so x(0) follows x(N-1)) which can result in sidelobes as well. The type of window used dramatically affects the resulting output spectrum, often reducing the high sidelobes at the expense of a wider mainlobe. The disadvantages of windows is the loss of information from the tapered spectral shaping which may not be representative of the signal spectrum, and a more difficult evaluation of the output image, i.e., the MSE cannot be used as a good measure of image quality. Windows have also been used successfully in the design of FIR filters [59]. Indeed, the weighted sinc interpolator of Chapter 3 was essentially a filter designed by tapering the sinc function

with a Hamming window.

Harris [49] has examined a number of windows and cataloged them according to peak sidelobe levels, mainlobe width, energy leakage, and several other parameters. The windows are also presented graphically, and examples of using windows in spectral estimation are given. Additional window designs and measurements are given by Geckinli and Yavuz [60] which favor the Hamming and Kaiser-type windows for spectral analysis and demonstrate the optimality of the truncated sinc window for an energy maximization criterion. Because SAR can be viewed as a spectral estimation problem, windows are examined which can improve sidelobe reduction and resolve point targets.

In this section, we briefly examine some of the windows which were used in the computer evaluations of 2-D interpolators. Previous work in optimal window design for irregularly shaped data set is also discussed.

4.1.1 Separable versus circularly symmetric windows

Probably the most common windows used in two-dimensional signal processing are separable. That is, they can be represented as

$$w(x,y) = w_1(x) \cdot w_2(y)$$

where typically, w_1 and w_2 are the same function, which is a good, one-dimensional window.

Huang [61] has shown that good circularly symmetric filters can be designed by rotating good one-dimensional windows, i.e.,

$$w_{2-D}(x,y) = w_{1D}(\sqrt{x^2 + y^2})$$

The circular weighting function has been using in optical SAR processing due to the nature of the spherical lens. It was also the weighting function chosen by ERIM for their processing of a rotating platform imaging system [62].

Unfortunately, when using windows with the DFT for spectral analysis, a window which is non-zero (although equal) at the endpoints will generate discontinuities when rotated, and thus produce higher sidelobes than the one-dimensional case. A Hamming window is one example. The two-dimensional circularly symmetric Hamming window shown in Fig. 4.1b displays the discontinuous boundary. This is an example of a window which is good for the one-dimensional and separable 2-D cases, but possesses a poor circular response. Experimental results in the next chapter demonstrate the inferior image quality. This may be corrected with a zero taper window such as the Hanning window which displays no such discontinuities (Fig. 4.2). The additional problem with the rotated window is the great information loss due to the relatively large area of zero weighting near the high frequency "corners." If the energy concentration is approximately zero for $x^2 + y^2 \geq r^2$, then a circularly symmetric window may be appropriate. It would work very well if the Fourier transform data had a circular region of support. However, spotlight mode SAR, in general, does not.

4.1.2 Optimal windows for SAR

Windows can also be designed which are optimal in a given sense. In [63], Papoulis listed several optimization criteria such as maximum energy concentration, specified zero crossings, minimum energy moment, and the minimum amplitude moment. The maximum energy concentration criterion generates a window which has maximal energy in a given range ($-\sigma$ to σ) and is solved via the prolate spheroidal wave functions (PSWF). Solutions, both continuous and discrete, of the PSWF are discussed by Slepian [64, 65, 66]. The specified zero crossings criterion is a generalized window of the form

$$w(t) = k(1 + 2\alpha \cos(at))\, p_\tau(t) \tag{4.1}$$

where

$$\alpha = \frac{-2 \sin(a\tau)}{2a\tau + \sin(2a\tau)} \tag{4.2}$$

where the zero crossings are specified by a. The Hamming window is a special case where the

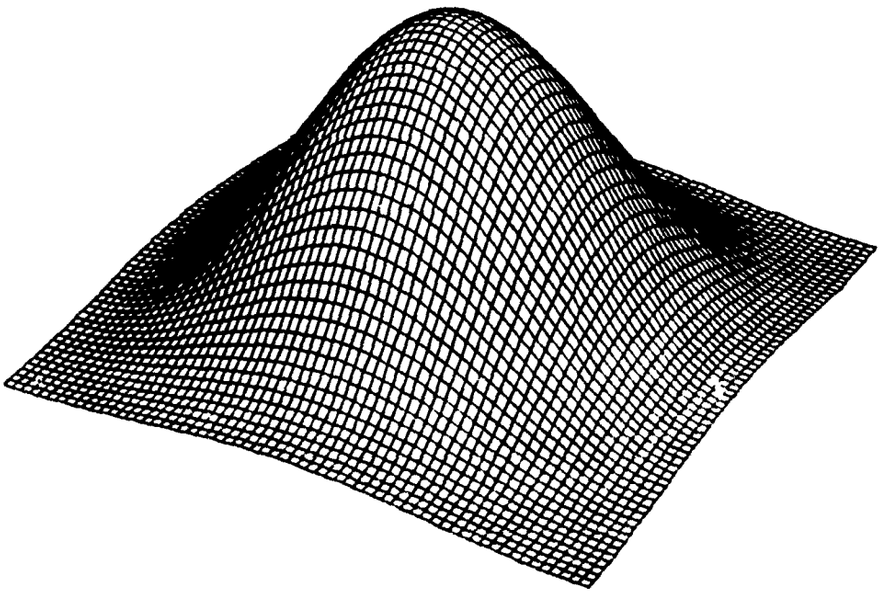


Figure 4.1a A Separable Hamming Window.

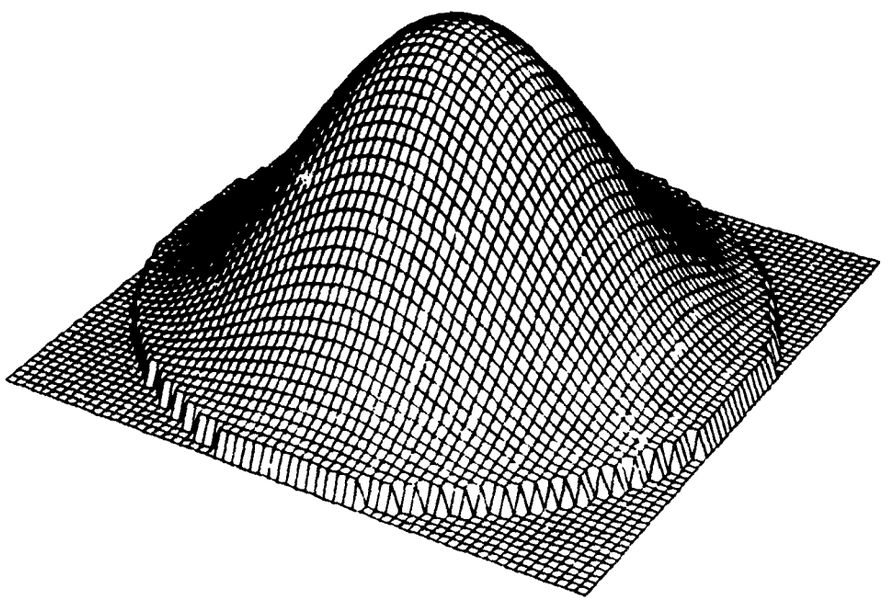


Figure 4.1b A Circularly Symmetric Hamming Window.

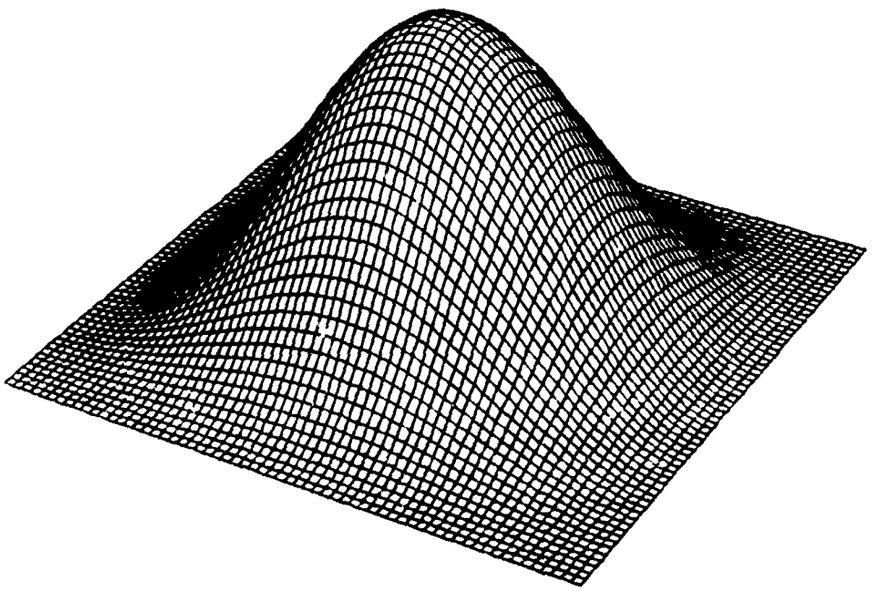


Figure 4.2a A Separable Hanning Window.

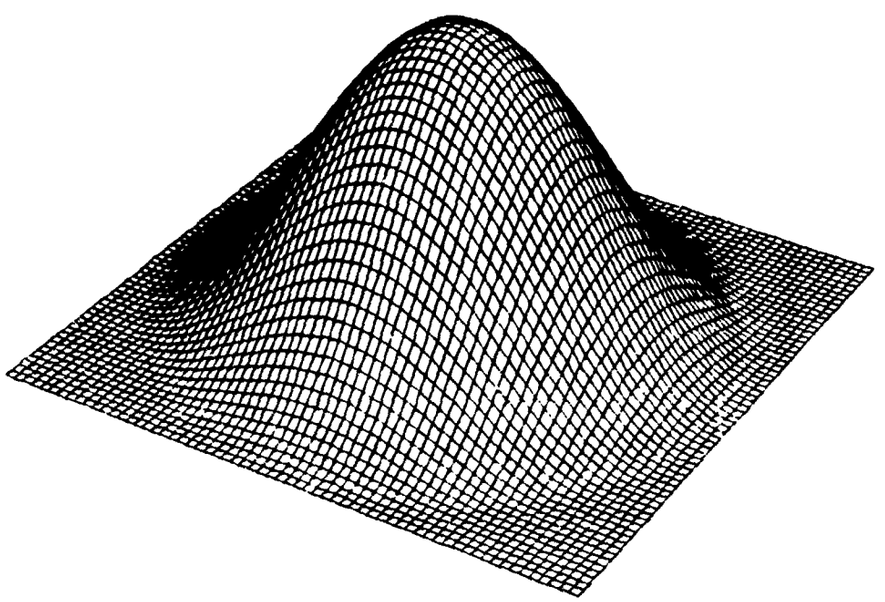


Figure 4.2b A Circularly Symmetric Hanning Window.

discrete window transform has zero crossing on the DFT sample points. This property makes the Hamming window useful for the interpolation evaluation problem since, a properly placed and sampled Fourier domain Hamming window will result in an image sampled in the nulls of the image domain sinc function. In this way, the interpolation error can be calculated independently of windowing effects. Several examples of circularly windowed data records are given in the experimental results.

The previous sections demonstrated the design of two-dimensional windows via a rather heuristic algorithm (i.e., take a good one-dimensional window and transform it into a two-dimensional window). However, neither method takes advantage of the shape of the region of known data. O'Brien [67] and Staehling [68] examine the question of optimizing the design of windows for irregularly shaped data regions. Since the spotlight mode SAR data lie on a toroidally shaped region, it is worthwhile to examine how these windowing strategies can be used prior to the FFT operation.

O'Brien investigated the use of an energy maximization criterion to generate windows which can be used on spatially limited images of irregular shapes, e.g., an L shape, a duel cone, and a circle. The solution is based on an iterative application of the time and frequency limiting operations found in the Papoulis extrapolation procedure [69]. The window design is formulated as an eigenvalue problem.

$$\lambda f(n,m) = B T f(n,m) \tag{4.3}$$

where

$B = W^{-1}BW$ is the bandlimited operator,

T is the spatial truncation operator,

B is the frequency-limited operator, and

W is the DFT matrix.

O'Brien goes on to demonstrate the convergence of the algorithm, and gives several examples

which show the types of windows designed for the simple geometric shapes given above.

Staehling used the Hamming window applied in directions orthogonal to the edges of the given data set (Fig. 4.3). The individual window arcs were expanded to meet the edge of the known data region. His results demonstrated better sidelobe control than the separable or circular case. This may be expected since the windowed data record becomes a function which gradually approaches zero.

The limitation of these algorithms is due to the lack of knowledge of the shape of a general data set. Also, the algorithms are numerical optimization procedures. Specifically, the data region shape must be known before design or application of the adaptive window design. If this *a priori* information is unavailable, then the separable window (somehow optimized for the full region) is the better and (computationally) simpler alternative. In the case of the toroidal

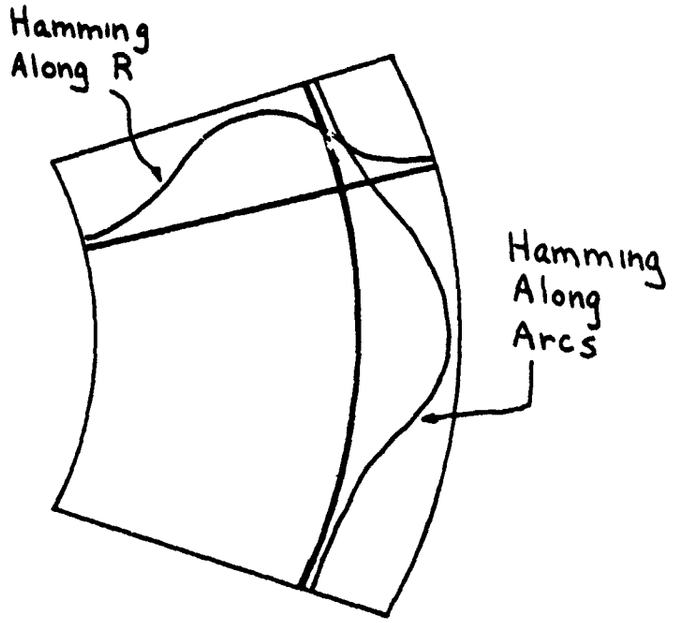


Figure 4.3 Application of the Hamming window to Orthogonal Edges.

data set, the previous strategies could be used only if the toroid were sampled on a rectangular raster prior to windowing with zeros surrounding the data record (Fig. 4.4). This, however, is precisely the problem faced in the interpolator. After the interpolation step, the window boundary is a moot point since the data are now on a rectangular raster and standard windows can be applied. Because the Hamming window allowed the placement of zeros at the image sinc zeros, it was used after the interpolation stage for the sidelobe reduction.

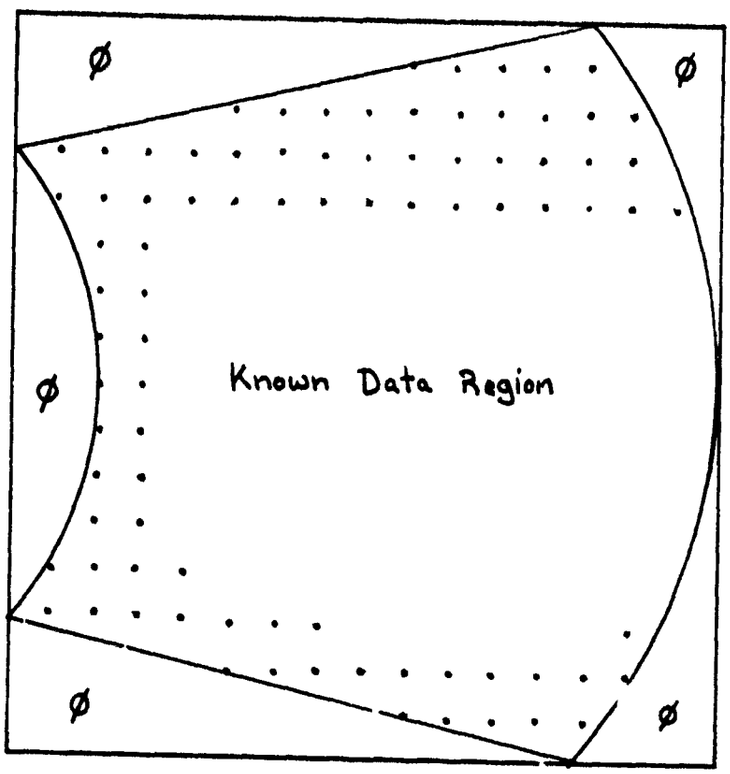


Figure 4.4 The Toroidal SAR Data Set on a Rectangular Raster.

4.2 The Limited Data Record and Extrapolation

This section briefly discusses the effects that a limited data set in the Fourier domain has on the image reconstruction. This is a ubiquitous problem for signal processing and is covered a great deal in the area of *extrapolation*. The data set that we are given extends over only a very small part of the Fourier domain - a piece of a torus. Outside this region, the data are non-zero (for spatially limited images, it cannot be zero), but in interpolating from one grid to another, it is often assumed to be so. Bandlimited extrapolation of the known signal set would seem to be worthwhile in SAR for two reasons: 1) the resolution of the system is proportional to the size of the Fourier piece so that by extending the data set, we can increase the system resolution; 2) when interpolating from one grid to another, the high-order interpolators will not "fall off" the sample grid near the edges. In practice, (2) is not as much a problem as would appear, because the size of a real system raster is several orders of magnitude larger than the interpolator kernel and errors would only occur near a few corners of the grid (Fig. 4.5). O'Brien [67] briefly examined the possibility of extrapolating the Fourier data set beyond the recorded region, but concluded with results consistent with Abend [70], who notes that bandlimited extrapolation has only limited use in spectral estimation.

4.3 Spurious Targets

It is well-known that interpolation errors in the Fourier domain lead to a noisy response in the spatial domain. The very nature of the transform operation predicts a global error in the spatial domain for singular errors in the Fourier domain. The following sections review and attempt to qualitatively characterize this *interpolation error effect* for tomography and SAR.

4.3.1 Munson-O'Brien analysis In [67] O'Brien examined the effect of interpolation in one domain on the transform domain, specifically looking at sample rate changes during the interpolation stage. The results help explain the presence of *spurious targets* in the output image.

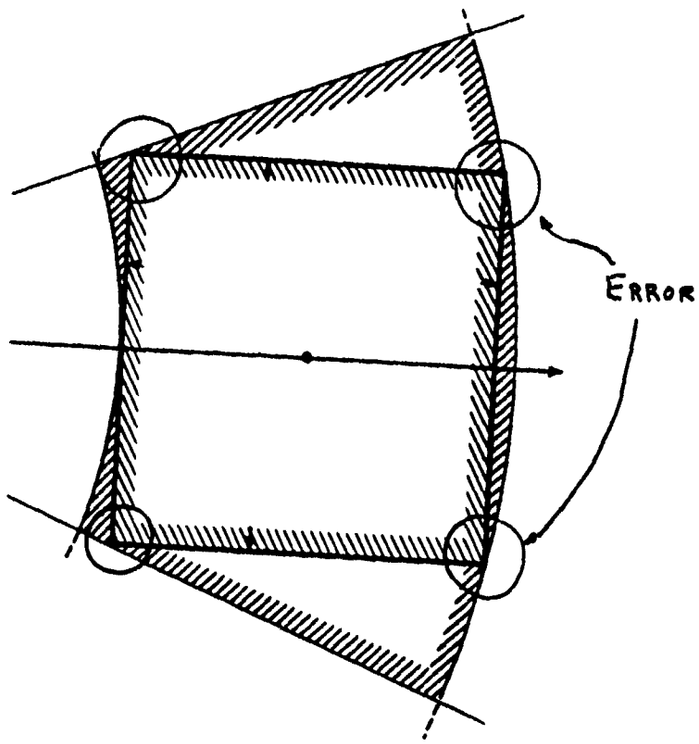


Figure 4.5 Areas where the Interpolation Kernel Falls Off the Edge.

In[16], Munson clarifies O'Brien's analysis of spurious targets. The result of[16], are presented briefly here for completeness.

The analysis can be simplified if the Cartesian-to-Cartesian (C-C) interpolation problem is reduced to one-dimension. Though the SAR interpolation problem is polar-to-Cartesian (P-C) rather than the simpler C-C geometry, the analysis can be used to predict where spurious targets may appear, since the section of the polar grid is approximately rectangular. This type of approximation may not be useful in the tomographic setting where the grids have markedly different shapes.

The sample rate conversion problem can be viewed as an analog interpolator followed by a sampler with the new period T'. In the case where the interpolation is performed in the Fourier domain, the process can be viewed as shown in Fig. 4.6, with the input sample set

available after the initial transform operation. The case and hat notation here is like that defined in the Chapter 2 discussion on the projection slice theorem. The imaging process performs the Fourier transform of f(t), into $F(\omega)$. $F(\omega)$ is then sampled at times nT to form F(nT). The sample rate conversion step can be thought of as a two step process of analog interpolation followed by a resampling operation. Because the interpolation step is inexact, it produces the continuous function $\hat{F}(\omega)$ which contains aliased copies of f(t). This is resampled as $\hat{F}(mT')$ and converted back to the time (spatial) domain as $\hat{f}(k)$ via the FFT operation. We would like to determine the relationship between $\hat{f}(k)$ and f(x). Note that $\hat{f}(k)$ is a sampled function, and the original function f(x) is continuous. The continuous interpolation kernel is h(t) with transform $H(\omega)$. It will be convenient to define the infinite sampling sequence $S_T(x) = \sum_{-\infty}^{\infty} \delta(x-kT)$ which is used in both the time and frequency domain. It is important to remember that the operations normally performed in the time domain (sampling, convolution, etc.) are now being done in the frequency domain, but the duality property of Fourier theory can be used to understand what is happening in the image (time) domain.

The output of the initial sampler (after the Fourier inversion step) is $F(nT) = F(\omega) \cdot S_T(\omega)$. This has the effect of convolution with the infinite impulse sequence $S_{\frac{2\pi}{T}}(t)$ in the time domain. Aliasing can result if T is insufficiently small. The sampled Fourier sequence is convolved with the continuous interpolation kernel, H(omega) to reconstruct an approximation to F

$$\hat{F}(\omega) = \sum_n F(nT)\, H(\omega - nT) \tag{4.4a}$$

$$= \{F(\omega) \cdot S_T(\omega)\} * H(\omega) \tag{4.4b}$$

In the time domain, this becomes

$$\{f(t) * S_{\frac{2\pi}{T}}(t)\} \cdot h(t) \tag{4.5}$$

The final sampling step multiplies the reconstructed function with the infinite sampling

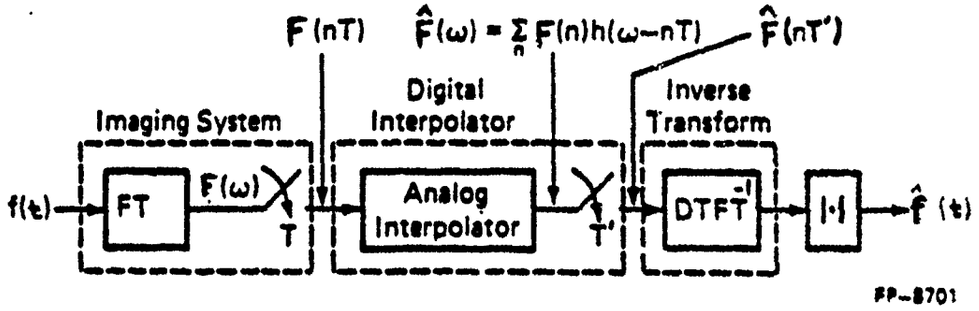


Figure 4.6 Sample Rate Conversion Model for Fourier Data Interpolation.

function, but with period T'

$$\left|\{F(\omega)\cdot S_T(\omega)\} * H(\omega)\right| \cdot S_{T'}(\omega) \tag{4.6}$$

and in the time domain

$$\left|\{f(t) * S_{\frac{2\pi}{T}}(t)\} \cdot h(t)\right| * S_{\frac{2\pi}{T'}}(t) . \tag{4.7}$$

It is precisely this second convolution after the h(t) *windowing* which accounts for spurious (aliased) targets to appear in the reconstructed image. It was shown in the interpolation chapter that the interpolation kernel must have a transform, h(t), which removes the $\frac{2\pi}{T}$ spaced copies of the spectrum (In this case, the copies are of the image.) If h(t) were bandlimited, the spurious targets would not appear. This would, of course, create an infinite length interpolation kernel.

4.3.2 Analysis for the nearest neighbor interpolator

Stark analyzed the nearest neighbor algorithm in terms of random jitter noise in the polar-rectangular interpolation stage to explain streak artifacts and high-frequency noise induced into a direct-Fourier reconstructed tomographic image [22]. He generates a noise model of the positional error in the transform domain as a uniformly distributed variable between -1/2 and +1/2. This corresponds to the position error induced from using the nearest neighbor which must be within 1/2 a sample point away (sample period T normalized to 1). This positional error "noise" is uncorrelated with the signal. Stark showed a relationship between the positional error *noise*, the Fourier transform of the original signal and the Fourier transform of the interpolated signal which demonstrates that the positional error *generates* noise which is correlated with the signal. He concludeed that the correlated noise translates into streaking artifacts in the spatial domain, as well as producing high-frequency amplification. The streak artifacts are also found in the SAR interpolation problem as smearing across the reconstruction.

Some insight into what is occurring can be gained by looking at the u,v error terms of each interpolated point for the nearest neighbor algorithm. Let E_u be the amount of error in the u-direction and E_v be the error in the v-direction (Fig. 4.7) in the Fourier plane. In general, E_u and E_v will both be functions of u and v. Let there be a point target at (u_0 , v_0). The Fourier transform is

$$F\left\{\delta(u-u_0,v-v_0)\right\} = F(U,V) = e^{-j(Uu_0 + Vv_0)} \tag{4.8}$$

Let (U,V) be the nearest neighbor to the polar sample $(\hat{U},\hat{V})$. Thus, $F(\hat{U},\hat{V})$ is used instead of the true F(U,V). Define the transform relationship

$$\hat{f}(u,v) = F^{-1}\{\hat{F}(U,V)\} \quad <=> \quad \hat{F}(U,V) \equiv F(\hat{U},\hat{V}) \tag{4.9}$$

Now, expanding $F(\hat{U},\hat{V})$ in a 2-D Taylor series

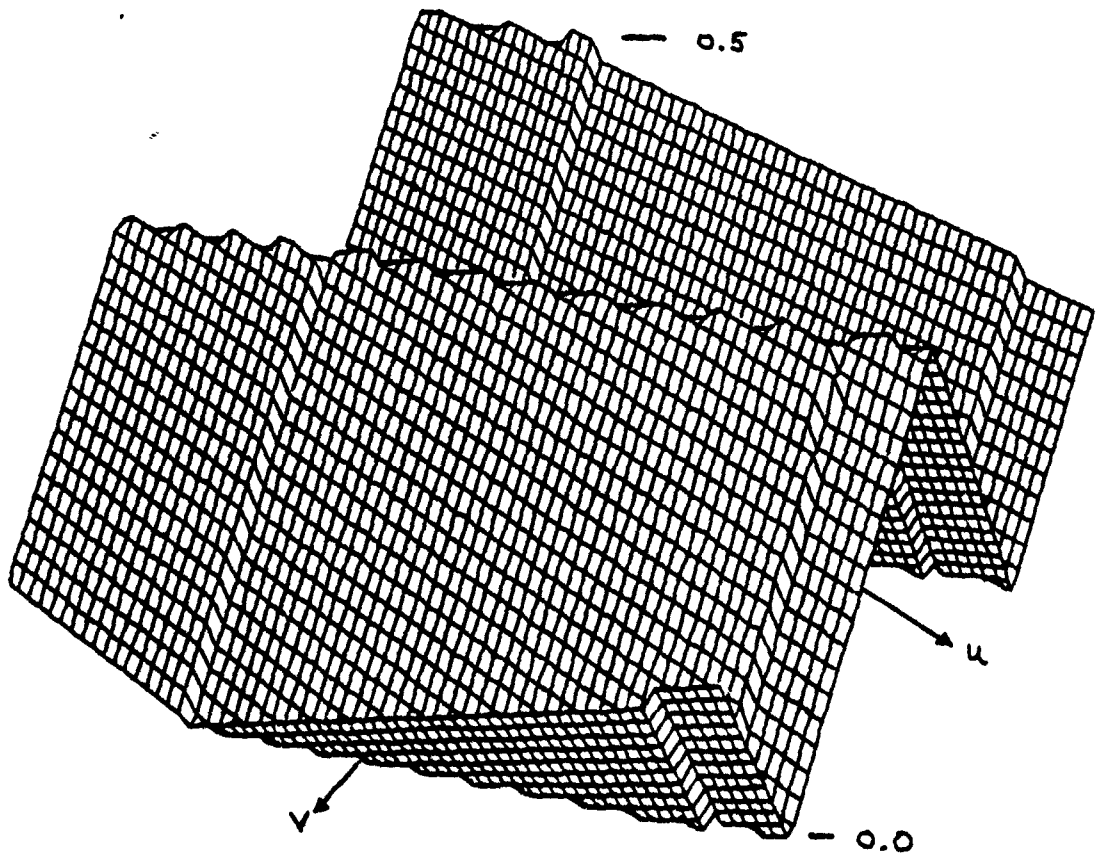


Figure 4.7a. Nearest Neighbor Position Error in U Direction.

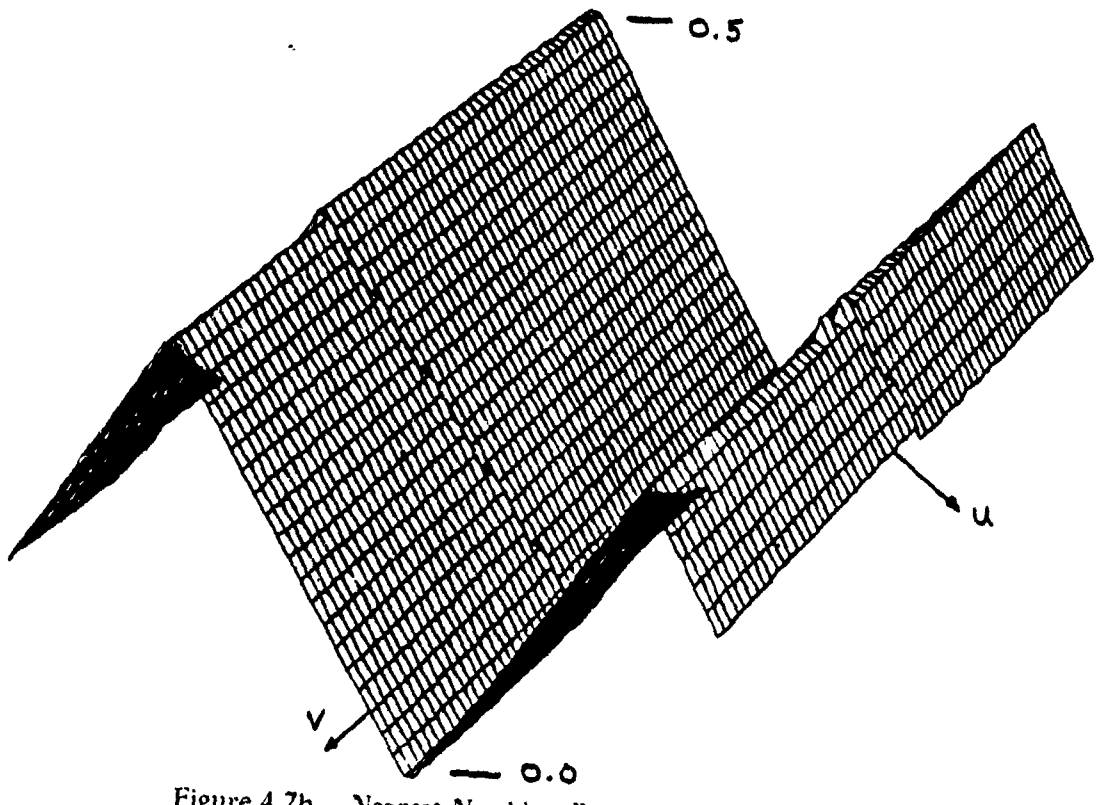


Figure 4.7b. Nearest Neighbor Position Error in V Direction.

$$F(\hat{U},\hat{V}) = F(U,V) + \frac{\partial F(U,V)}{\partial U}(\hat{U} - U) + \frac{\partial F(U,V)}{\partial V}(\hat{V} - V) + \text{higher order terms.} \tag{4.10}$$

But

$$E_u(U,V) = (\hat{U} - U) = F_{2\text{-}D}\{e_u(u,v)\} \tag{4.11a}$$

$$E_v(U,V) = (\hat{V} - V) = F_{2\text{-}D}\{e_v(u,v)\} \tag{4.11b}$$

is the transform of the positional errors for each coordinate which depends on (u,v). The partial derivatives of F are

$$\frac{\partial F(U,V)}{\partial U} = ju_0 F(U,V) \tag{4.12a}$$

$$\frac{\partial F(U,V)}{\partial V} = jv_0 F(U,V) \tag{4.12b}$$

Thus, (4.10) becomes

$$F(\hat{U},\hat{V}) = = F(U,V) + ju_0 F(U,V) E_u(U,V) + jv_0 F(U,V) E_v(U,V) \tag{4.13}$$

where the higher order terms have been dropped. The spatial domain equivalent is

$$\hat{f}(u,v) = f(u,v) + ju_0 f(u,v) ** e_u(u,v) + jv_0 f(u,v) ** e_v(u,v) \tag{4.14}$$

Equation (4.14) shows that the point target position and 2-D error surface are related to the image reconstruction though a 2-D convolution, and that the error surface is proportional to the position of the point target u_0 and v_0.

Now as a special case, suppose $E_u(U,V)$ and $E_v(U,V)$ are 2-D periodic with fundamental frequencies (α_u, β_u) and (β_u, β_v). Then:

$$E_u(U,V) = \sum_i \sum_k a_{ik}\, e^{j(i\alpha_u U + k\alpha_v V)} \tag{4.15a}$$

$$E_v(U,V) = \sum_i \sum_k b_{ik}\, e^{j(i\beta_u U + k\beta_v V)} \tag{4.15b}$$

and

$$\hat{f}(u,v) \approx f(u,v) + ju_0 \sum_i \sum_k a_{ik}\, f(u + i\alpha_u, v + k\alpha_v) + jv_0 \sum_i \sum_k b_{ik}\, f(u+i\beta_u, v+k\alpha_v) \tag{4.16}$$

Equation (4.16) represents the special case when the error surfaces in Fourier space are periodic. If the sampling frequencies in the input sampling array are approximately constant, as in the case where only sub-patches are used in the reconstruction, then the interpolation will be from a near-rectangular grid to an exact rectangular grid, which is perhaps rotated. The error surfaces in this case are periodic triangle waveforms with an orientation that determines α and β (Figs. 4.7a and 4.7b). By knowing the relative geometric orientations of the grids, the resultant smear pattern may be predicted.

Equation (4.16) also shows that the smear magnitude is proportional to the displacement of the point target from the origin. Points farther away from the patch center will cause greater smear. This is similar to the effect caused by the quadratic phase term in the SAR phase equations (Chapter 2).

The error surface directional vectors defined as $\vec{\alpha} = (\alpha_u, \alpha_v)$ and $\vec{\beta} = (\beta_u, \beta_v)$ describe the smearing direction and depend on the error surface orientation. The input polar grid displays a spatially varying period and orientation which is slowly modulated with respect to the rectangular grid. Although these error surfaces may not exhibit 2-D periodic behavior in the general case, for the near rectangular SAR geometry, and rectangular to rotated-rectangular grid geometries, they are approximately periodic.

4.4 Alternative Sampling/PRF Strategies

The previous sections have dealt primarily with the standard polar input raster. Modifications to the sampling and/or radar pulsing mechanism allows other sampling rasters to be used with greater efficiency and reduction of error. This concept was presented in a tomographic setting by Mersereau and Oppenheim [71] through the use of concentric square sampling. The resulting image reconstruction is significantly improved because interpolation is only done in one direction. Simplifying the interpolation geometry to one-dimension also reduces the computation of the interpolation stage, both in locating the grid points and performing the filtering/sample rate changes. Of course, this type of sampling strategy requires a much more sophisticated sampler. In SAR, alternate grids may be designed to take advantage of a smart sampling device,as well as a programmable pulse repetition frequency (PRF) generator. The keystone grid is rather well-known, but more as an intermediate interpolation grid rather than an input raster. Another, is a polar-like grid with equi-PRF spacing. The range samples are uniformly sampled for each return signal, and the pulses are transmitted at a uniform rate (the PRF is constant) as the radar moves past the terrain at a constant velocity. This results in a range lines which are unequally space in θ. However, the intermediate (keystone) grid produced by the separable algorithms will have equally spaced samples in the azimuthal direction.

4.5 Keystone with Smart PRF

If the incoming signal is sampled on the keystone grid shown in Fig. 3.6, then the complex two-dimensional interpolation reduces to one-dimension. This greatly reduces interpolation error and computation time as shown by the computer simulations of Chapter 5. It also allows for a novel interpolation-Fourier transform operation in one step - the *Chirp Z-transform*.

4.5.1 The chirp z-transform algorithm

An N-point DFT of a sequence can be thought of as sampling the z-transform of a sequence uniformly around the unit-circle in the z-plane. The chirp z-transform is widely known as a generalized DFT algorithm that computes M samples of the z-transform along an arbitrary spiral contour in the z-plane. Once more, N and M need not be the same as in the DFT, nor do they have to be highly composite for implementation with an FFT. The chirp z-transform is based on using the FFT to perform fast convolutions of the input sequence with a chirp, or frequency ramped, signal. The algorithm is explained in its entirety by Rabiner *et al.* in [72] of which a distilled version is found in [73]. It is used in pole enhancement, narrow-band frequency analysis, bandlimited interpolation, and arbitrary radix DFT computations. Although the most general form of the chirp z-transform allows for computation on an arbitrary spiral contour in the z plane, this application uses it to simply perform an M point DFT with an N point sequence, M and N being different. This means that the contour reduces to the unit circle in the z-plane with the output sequence beginning at a non-zero phase angle.

The M point chirp z-transform of the N point sequence x(n) is given by

$$X(m) = \sum_{n=0}^{N-1} x(n)\, A^{-n} W^{nm} \tag{4.17}$$

where $A = A_0 e^{j2\pi\theta_0}$ specifies the output start angle and radius, and $W = W_0 e^{j2\pi\phi_0}$ specifies the contour curvature, and output sample spacings as shown in Fig. 4.8. In the SAR case, the output contour is still on the unit circle, although it may be located at the arbitrary angle $2\pi\theta_0$ and extend to $2\pi(M-1)\phi_0$ where ϕ_0 is the output sampling angle (Fig. 4.9).

Through some algebraic manipulation, (4.17) can be reduced to a 3-step process of

(1) Ramping (multiplying by a frequency swept sequence) the N point input sequence by $A^{-n} W^{n^2/2}$.

(2) Convolving this result with $W^{-n^2/2}$.

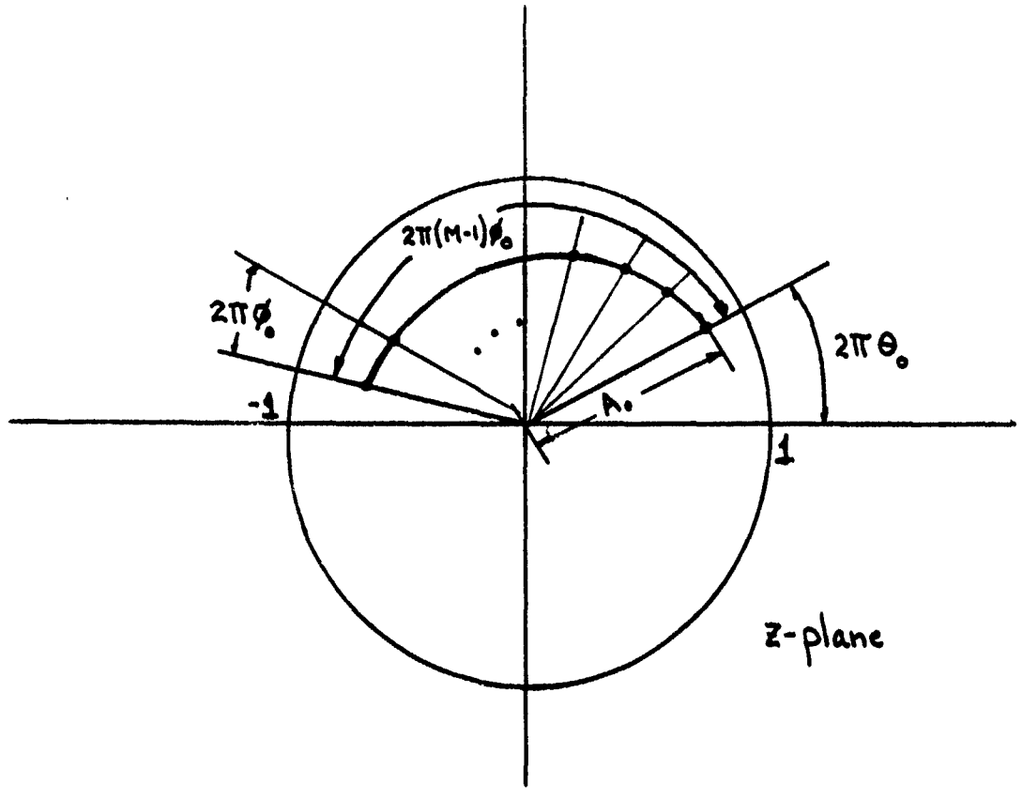


Figure 4.8 The Contour Represented by A and W in the Z-plane.

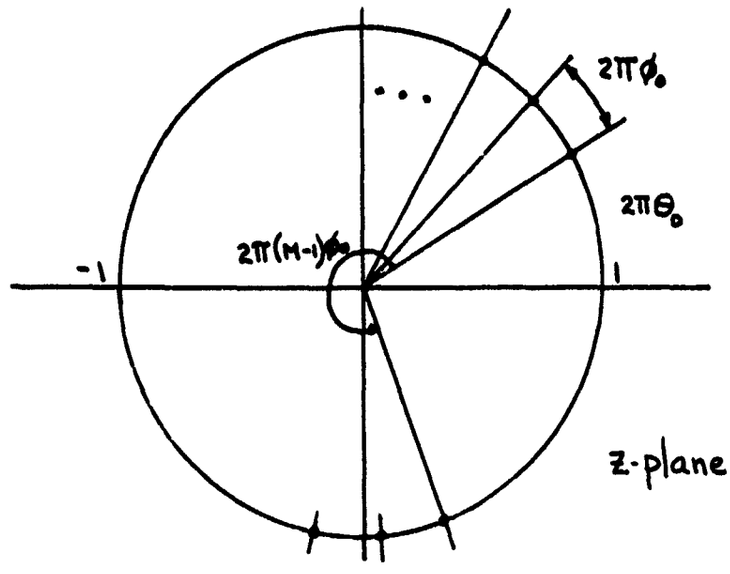


Figure 4.9 The Chirp-Z Transform Contour Lying On The Unit Circle.

(3) Deramping the convolved sequence with $W^{n^2/2}$.

This is illustrated in Fig. 4.10.

The normal convolution can be done with an FFT if the sequences are properly padded with zeros to avoid wrap around (circular convolution). Unfortunately, the zero padding increases the FFT size to N' which is the first power of two greater than N + M. This can lead to much higher computation time for small N, and make the chirp-z algorithm complexity disproportionately high compared to the other interpolation algorithms. It must be remembered, however, that the chirp-z approach folds the Fourier domain inversion step into the interpolation stage so that comparisons must be made with the 2-D IFFT step added to the complexity measure. This becomes less of a problem as N is increased, since the FFT time is order $N\log_2 N$.

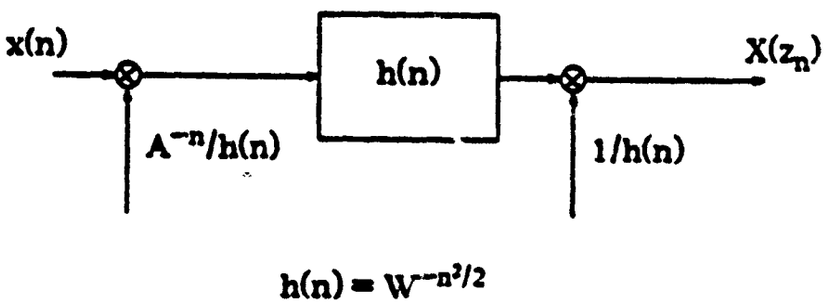


Figure 4.10 The Chirp Z-transform as a Pre- and Post-Multiplied Convolution.

In the SAR geometry, the Keystone sampling and smart PRF place data points on straight azimuth lines with sample points equi-spaced on any particular azimuth line (though the spacing changes from azimuth line to azimuth line). The chirp-z algorithm is applied to each keystone azimuth line placing the output points on the rectangular grid. A standard FFT is performed in the orthogonal direction. Here, $A = \exp(j\theta_0)$ and $W = \exp(-j\phi_0)$, where θ_0 is the angle of the first spectral sample and ϕ_0 defines the output sample spacing.

The number of input points N for the chirp-z algorithm is arbitrary, and the number of output points M is equal to the rectangular raster size - 1024 for the total image, 64 for the computer simulations. Each azimuthal input line is windowed (Hamming) prior to the chirp z-transform. Note that as the azimuth line index increases with range, the input sample spacing increases and the window size will increase. This is a modification of the warped windows discussed by Staeling and O'Brien. The range line window is applied prior to the secondary FFT stage.

The input data set lies on a trapezoidal raster and the output of each chirp z-transform must be correctly phased between azimuth lines. This can be done by multiplying each output point by a linear phase term that changes from azimuth line to azimuth line

$$\exp\{jv\,(u\tan\theta)\}$$

Because the chirp z-transform is a *forward* transform (which is followed by a *forward FFT*) and the data set is already in the Fourier domain, the resulting image must be time reversed, i.e., rotated 180 °, to correct the effect of a double forward transform since

$$FT\left\{FT\{f(x,y)\}\right\} = f(-x,-y)$$

4.6 Polar Format with Equi-PRF

A slight modification to the PRF will place the samples on an equi-PRF grid which appears like a polar grid, but with angular increments that are not constant (Fig. 4.11). This has the

advantage that the separable algorithms in Chapter 3 which generate an intermediate keystone grid will have equi-spaced sample points along the azimuth lines (just as the smart-PRF keystone grid in the previous section). This simplifies the interpolation step in the second stage of the separable algorithms (weighted sinc, spline) and thus reduces computation.

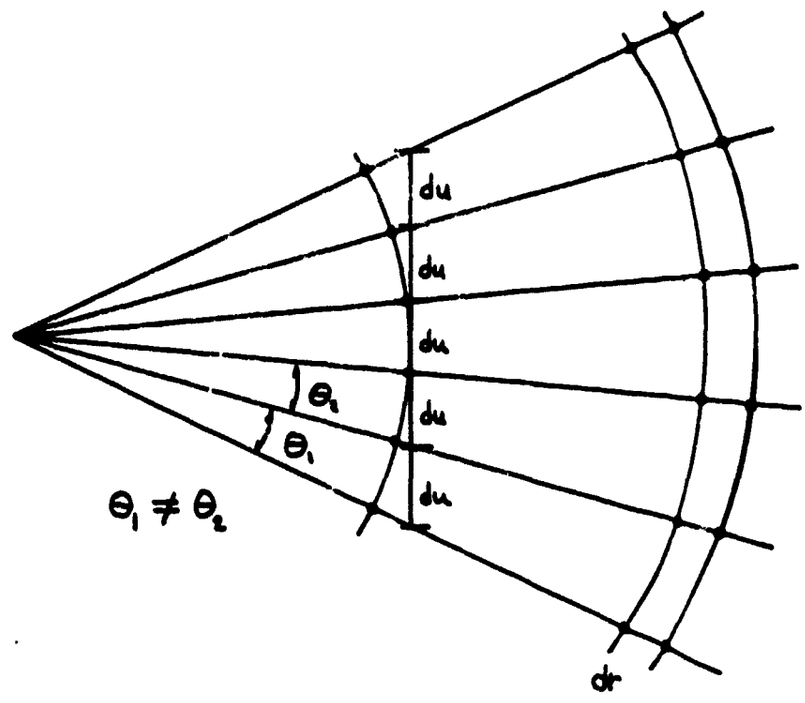


Figure 4.11 Equi-PRF Polar Grid.

CHAPTER 5

EXPERIMENTAL EVALUATIONS

The previous chapters presented the theory of interpolation for signal reconstruction in both spatial and frequency domains. This chapter applies this developed theory to the original problem of image reconstruction in synthetic aperture radar.

The following sections describe the approach taken for the computer experiments, the simplification to the SAR model used, and the results of the various interpolators. A novel measure of image reconstruction is also presented here. Because the number of parameters is large (Fourier section of interest, point target location, data record window, interpolator type, and interpolator order) only a subset of image plots is given. A more complete set of interpolator performance results are given in tabular form (Tables C.1 to C.12) in Appendix C.

5.1 Interpolation Model

In the evaluations, we assume a look angle, θ_{max} of 3°. This is the angle over which all of the data are collected. The rectangular array to be interpolated onto is inscribed within the two arcs R_{min} and R_{max}, which are proportional to the chirp bandwidth defined in Chapter 2. The input polar sample array is assumed to have a size of 1024 by 1024 points with 1024 equally spaced samples along each range line R_{min} to R_{max} and 1024 samples along arcs from $-\theta_{max}/2$ to $+\theta_{max}/2$.

A program to generate SAR data from the equations of Chapter 2 was written, allowing each of the many geometric and system parameters to be adjusted. While (2.26), represented an exact point target response, the smearing caused by the quadratic phase term obscured the effects of the interpolator, and thus were abandoned for a newer, simpler model: the Fourier transform of a spatially offset impulse function. A single, ideal point target was placed at (x_0,y_0) and the corresponding Fourier transform, $e^{-j(U x_0 + V y_0)}$, was sampled on the uniform polar grid. Allowable target positions which did not produce aliasing ranged from -32 to +31

in the spatial domain. We used (-23,24) as the point target position in the evaluations presented here. This stressed the interpolator by including high-frequency components while allowing the filter response to have a finite-width transition band. High frequencies stress the interpolator because a higher degree of variation is more difficult to reproduce with simple kernels.

Because our computer resources did not permit processing such a large array, we chose to examine interpolator effects on various 64 × 64 subarrays of the full 1024 by 1024 grid. Each 64 × 64 subarray is numbered from 1 to 16 along the U-axis, and from 1 to 8 along the V-axis. For example, subarrays (2,8) and (16,1) are shown in Fig. 5.1. Symmetry about the U-axis eliminates the need to study the interpolator for $v < 0$. The subarrays (2,8), (10,5), (16,1) and (16,8) were studied, since they represented various sections of the full array that would have unique properties, i.e., rotation, maximal sample rate change, average performance, etc. The input data region of each subarray was extended in both range and azimuth by 15 sample points in each direction (making the available data 94 by 94 samples) so that the higher order interpolators did not "fall off" the edge of the input. Since the original 1024 by 1024 grid had only a very small set of points where this occurs, this data extension was justifiable for the subarrays.

5.2 A Novel Figure-of-Merit

Evaluation of the interpolators is a difficult task, especially since radar data interpretation tends to be subjective. The figure of merit that we have used is called the Multiplicative Noise Ratio (MNR), which is very similar to the more commonly known "integrated sidelobe ratio" frequently used for one-dimensional radar evaluation [1]. Because the rectangular grid is weighted with a separable Hamming window, an ideal point target is spread 3 samples in each dimension U and V. In the following experiments, the mainlobe of the response is defined as the 5 × 5 square centered on the peak response. The MNR for the reconstruction is then defined

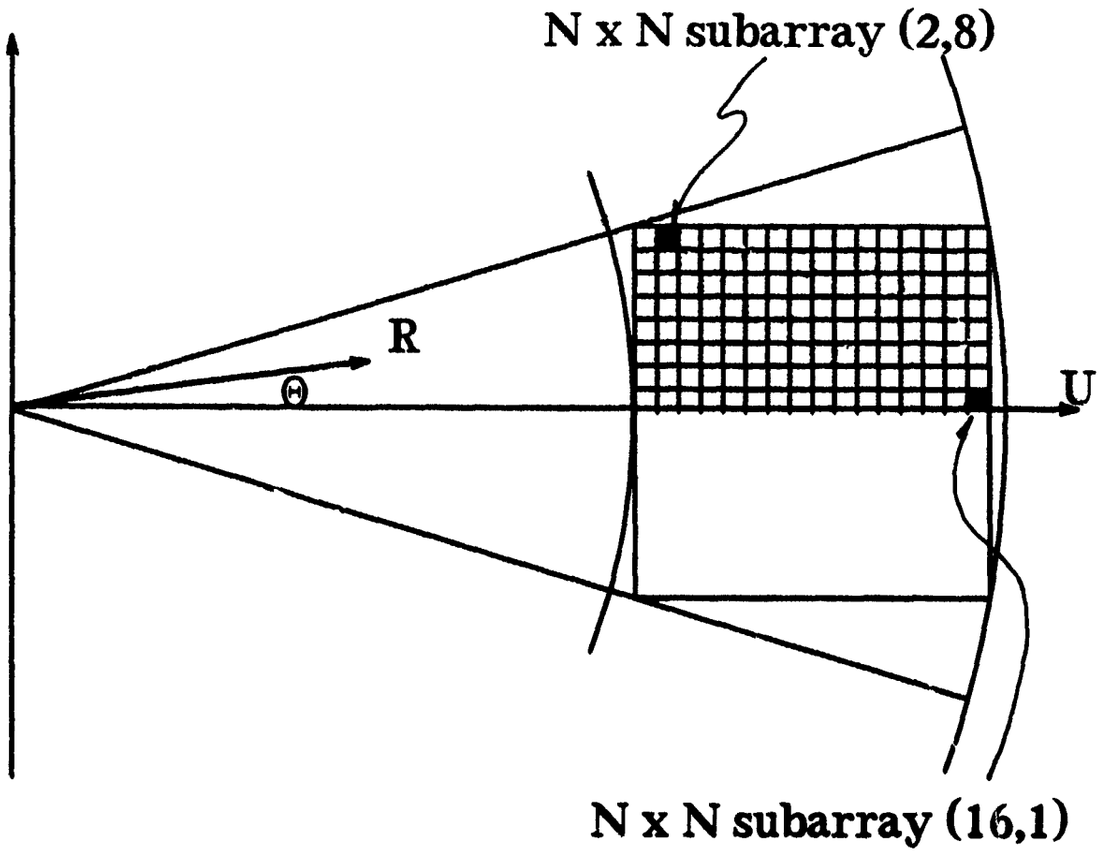


Figure 5.1 Geometry of Fourier Domain Subarray Locations.

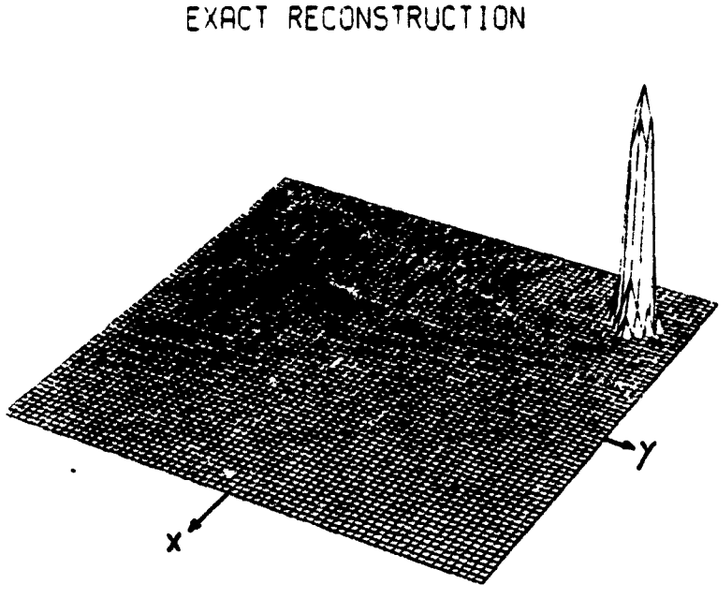


Figure 5.2 Exact Reconstruction for a Target at (-23,24).

as

$$\text{MNR} = 10 \cdot \log \left| \frac{\sum(\text{magnitude of the points outside mainlobe})^2}{\sum(\text{magnitude of the points inside mainlobe})^2} \right|$$

Figure 5.2 shows a perfectly reconstructed ideal point target at (-23.0,24.0) with a calculated MNR of -48.13 dB. It was produced by generating target data directly on a rectangular grid and bypassing the interpolating stage during reconstruction. The floor for *all* point target response plots has been set to -60 dB relative to the peak value. Note, in Fig. 5.2, the effect of the Hamming window on the ideal response. The point target has widened and the rest of the samples fall on the nulls of the sinc (the response for a point target on a limited record). If the target is moved to a position halfway between sample coordinates, the sinc response is then sampled *on* the peaks of the sidelobes, resulting in a badly distorted response and an MNR of -28.5 (Fig. 5.3). This demonstrates that the MNR calculations are valid only for targets which are positioned *directly* on the sample points. If the interpolator caused the target to move only slightly, the sidelobes become visible and the MNR increases misleadingly. It is interesting to note that the MNR can produce values which apparently differ from a subjective view. Sometimes, one reconstruction will *look* better than another, but the MNR value is worse (more positive). This is usually the result of a very wide target response in which the samples directly outside the true 5 by 5 mainlobe are summed in the denominator, causing an increase (more positive) in the MNR. The results are thus obtained as follows:

(1) Generate a sampled polar array by sampling the Fourier Transform of the ideal target.

(2) Interpolate to the rectangular grid of interest (1,1) to (16,8).

(3) Window the resulting 64 × 64 rectangular data subarray with a separable Hamming window.

(4) Calculate the IFFT.

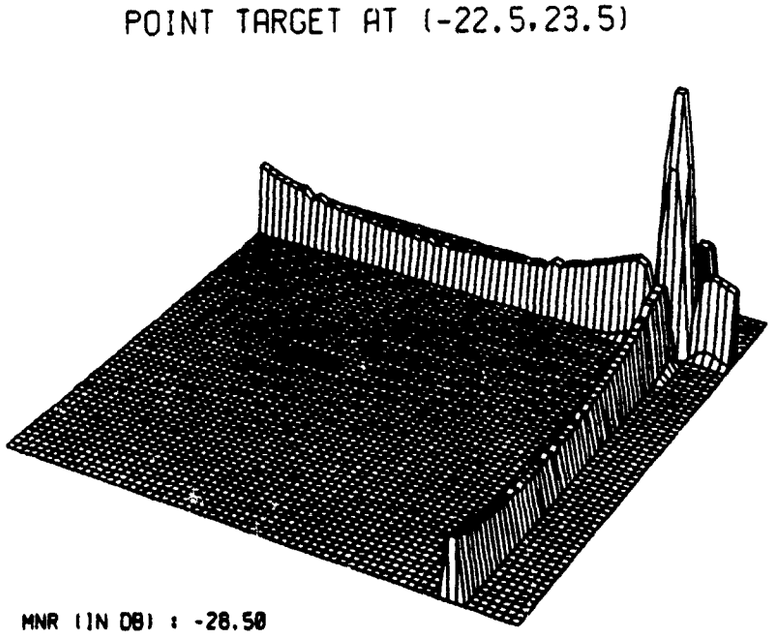


Figure 5.3 Exact Reconstruction for a Target at (-22.5,23.5).

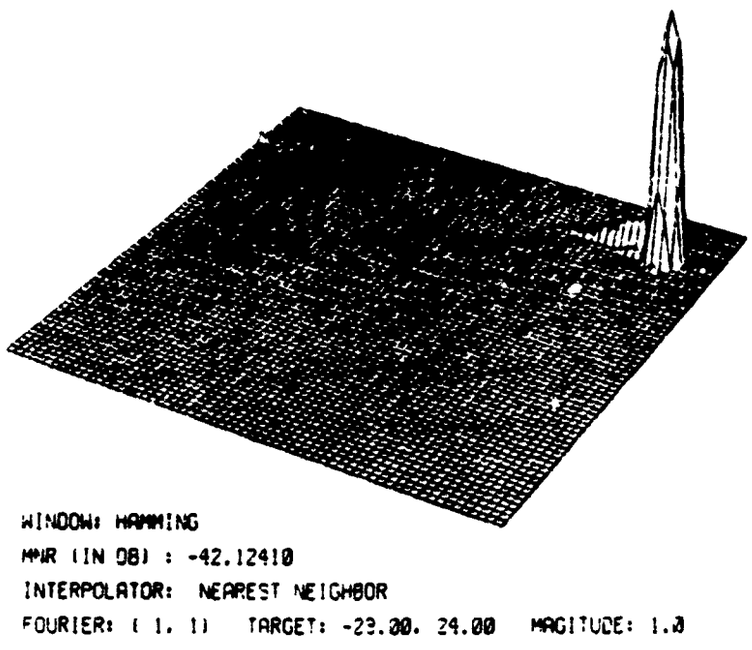


Figure 5.4 Nearest Neighbor Interpolator at (1,1).

(5) Calculate the MNR of the spatial domain image.

(6) Display the $\log_{10}$ of the squared magnitude of the reconstruction (the power) with a floor of -60 dB.

The program *evaluate* calculates the image MNR and locates the largest (magnitude) twenty points on the grid to determine the positions of the spurious peaks and false targets. The dB levels of these points relative to the detected peak are also calculated.

5.3 Algorithm Complexity

This work focused on improving the SAR reconstruction quality, while reducing the interpolation processing time, i.e., reducing the computational resources required to perform the algorithm. The required resources are specified by what we have termed "complexity" - a measure of how complex (in terms of mathematical operatations) the algorithm is to perform. Algorithm complexity is a difficult issue, because the algorithms presented do not offer an order of magnitude increase in processing speed, but typically improve speed by a factor in the range of 2 to 10. Also, the results are dependent on the order of the interpolator.

Crochiere and Rabiner[35] use the notion of Multiply-ADditionS or MADS/sec when optimizing the decimation/interpolation stages for a sample rate converter. This was used because the final design was a simple time-invariant FIR filter with real coefficients which has only multiplies and adds. Neither complex arithmetic or divisions are required, nor transcendental function evaluation (sine, square root, etc.). They also make the assumption that an add takes the same amount of time as a multiplication. This may not be totally unfounded, as in the case when all arithmetic (necessarily integer) is done with a look-up table and requires only one clock cycle. Real multiplies, however, typically require more time than real adds, and real divide operation is more expensive than a real multiply (although no divides are performed in the simple 1D FIR filters).

Here, the algorithm complexity is calculated by computing the number of operations of a given type that occur in the interpolation software. Though this measure would seem to be based on the particular programming practice, it takes into account many of the operations normally ignored by "order of complexity analyses," such as calculation of the coordinate spatial position and the evaluation of trigonometric functions. The complexity was determined by diagramming program loops and generic operations and identifying function calls. Trigonometric and transcendental function calls were given their own unit of cost measure so that the complexity would not be skewed by a particular implementation, such as look-up table, power series expansion, or a table-interpolation scheme. Thus, the cost to evaluate the sin function is represented as C_{trig}, the cost to compute a square root is C_{sqrt}, the cost to compute a complex exponential is C_{exp}, and the cost to generate a sinc value is C_{sinc}. The cost of a real add or subtract is $C_{a/s}$ and the cost of a real multiply or divide is $C_{m/d}$. Although it may seem that this is a crude estimate of algorithm complexity, it has been found through our experience to be reasonable. The complexity calculation was based on the interpolator alone, and did not include the windowing, file reading/writing, or post processing times (array permuting, magnitude detection, etc.). The FFT stage has been included in the complexity analyses because the chirp-z algorithm performs both interpolation and FFT as one, inseparable operation.

The computation time for the interpolation set was measured for each algorithm with varying orders and different input grids. Because the VAX CPU timing facility is system load dependent (the measured cpu time is a function of the load of the system), an average run time value was computed over all of the runs for a particular grid and interpolator. This value is used to plot the algorithm times and to compute another performance measure - the IMNRI/CPU ratio. This ratio is useful to describe the quality/cost of an interpolator. Higher ratios indicate a better reconstruction for the amount of computation used. It is only useful to compare interpolators within a given subarray, since the MNR ratios vary dramatically between subarrays with similar order interpolators.

5.4 Polar Grid

The polar grid input array represented the most well known data geometry. The Fourier piece (1,1) is the subarray which is closest to the output grid in both sample spacing and orientation. This leads one to believe that the nearest neighbor interpolator should perform quite well here. In fact, it does very well, as shown in Fig. 5.4 with an MNR of -42.12. This is only 6 dB away from the ideal response.

In subarray patch (2,8), the nearest neighbor does not perform nearly as well (Fig. 5.5). With an MNR of -5.64, the reconstruction is extremely noisy and contains many streaky artifacts. It is interesting to note that the nearest neighbor produces a streak pattern which is similar to the artifacts described by Stark [22] in his nearest neighbor analysis. It can also be explained in terms of the nearest neighbor analysis presented in Chapter 4. The rotated grid in (2,8) generates an almost periodic positional error surface which has an orientation related to

TABLE 5.1 Evaluation Results for Fourier Piece (2,8) on Polar Grid.

Target Position	Interp.	Parameter	MNR (db)	CPU time (seconds)	|MNR/CPU|
-23,24	NN		-5.64	3.28	1.72
-23,24	G-ID	(0,1)	-14.56	19.82	0.73
-23,24	G-ID	(0,2)	-18.41	5.75	3.20
-23,24	G-ID	(0,3)	-17.00	22.57	0.75
-23,24	wsinc	2	-18.92	17.7	1.07
-23,24	wsinc	4	-24.50	28.4	0.86
-23,24	wsinc	6	-30.37	41.3	0.74
-23,24	wsinc	8	-36.53	53.0	0.69
-23,24	wsinc	10	-42.65	61.7	0.69
-23,24	wsinc	12	-46.88	70.6	0.66
-23,24	wsinc	14	-48.06	82.7	0.47
-23,24	wsinc	16	-48.24	94.7	0.51
-23,24	B-spline	-0.25	-23.01	17.22	1.34
-23,24	B-spline	-0.50	-25.17	17.22	1.46
-23,24	B-spline	-0.75	-27.62	17.22	1.60
-23,24	B-spline	-1.00	-30.46	17.22	1.77
-23,24	Spline (IMSL)		-32.41	27.7	1.17

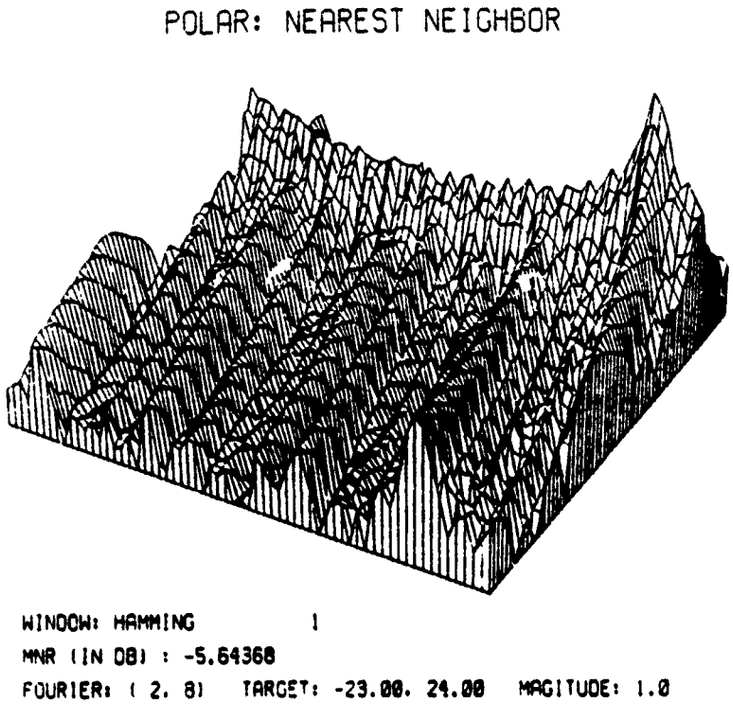


Figure 5.5 Nearest Neighbor Interpolator at (2,8).

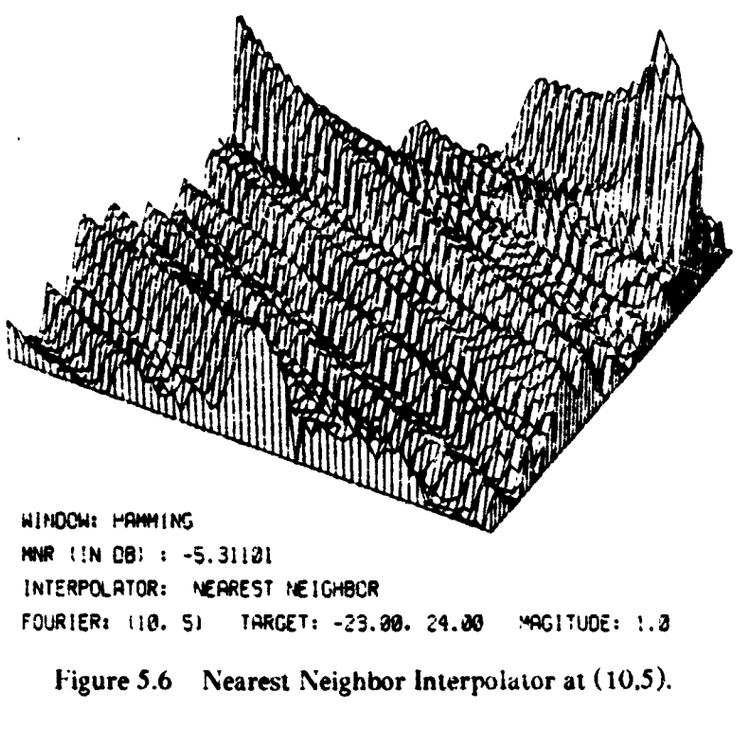


Figure 5.6 Nearest Neighbor Interpolator at (10,5).

the relative input/output grid angles. The same type of smearing is present when the target is moved to (1,-15) (Fig. 5.7). The MNR is a little better at -10.63, because the Fourier data is more slowly varying, but the reconstruction noise obscures the image. Finally, in the (16,1) subarray, the nearest neighbor interpolator generates many spurious targets due to the change in sample rate (Fig. 5.8). The azimuthal rate change at this position in space is approximately 1.05, and the Munson-O'Brien equations [16] predict spurious targets along the axis of rate change (the predicted position is 27.2, the measured position is 27) with magnitudes proportional to the interpolator frequency response beyond the cutoff frequency. Since the nearest neighbor interpolator transform is a sinc, the sidelobes are very high, and thus, produce the great numbers of high spurious targets. It is difficult to measure the angular orientation of these error surfaces (they are *not* at the polar grid angle) but the angled smearing is quite clear (Fig. 5.5). The same type of smearing is present in the nearest neighbor reconstruction of subarray (10,5); however, it is in a different direction (Fig. 5.6).

Figure 5.7 also shows that the use of the FFT as an approximation to the Fourier transform has produced a reconstruction which is apparently one period of a 2D periodic signal. The other reconstructions reflect the same conclusion. This is, of course, expected, since the finite record sampling operation implicitly *creates* a periodic signal.

The inverse distance (ID) interpolator performed better than the nearest neighbor (NN) in all subarrays, based on the MNR values. In (2,8) (Fig. 5.9), there is a marked decline in the MNR: -14.56 dB (ID) compared to -5.64 dB (NN). Compare this with the inverse distance squared (ID^2) reconstruction of Fig. 5.10. The MNR of -18.42 suggests that the ID^2 is slightly better than ID, and furthermore, the interpolation time is 3.45 times faster. This is a significant improvement overall as seen by the |MNR|/CPU ratio in Table 5.1.

If the generalized inverse distance interpolator is used by including the next 16 nearest data points, both ID and ID^2 reconstructions are poorer (Figs. 5.11 and 5.12, and Tbl. 5.1). Increasing the *order* of the inverse distance interpolator beyond 2 did not seem to improve the

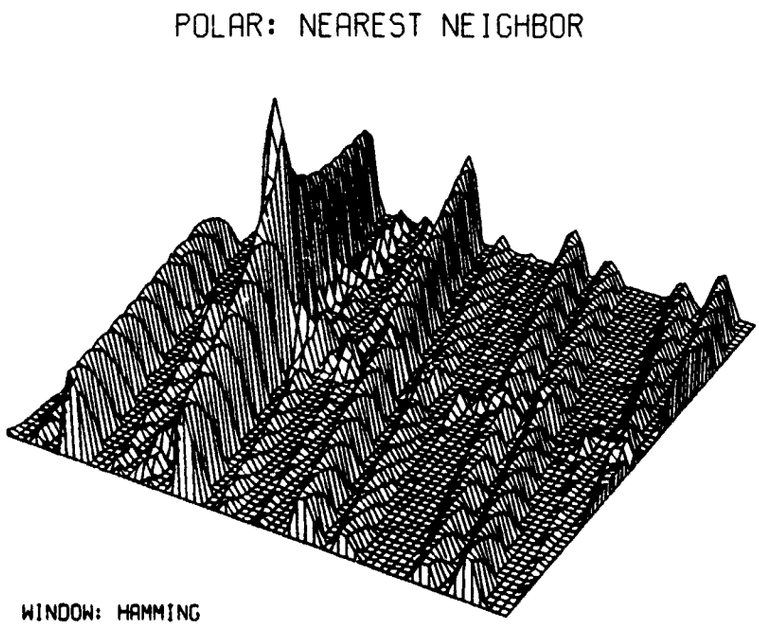


Figure 5.7 Nearest Neighbor Interpolator at (2,8), target at (1,-15).

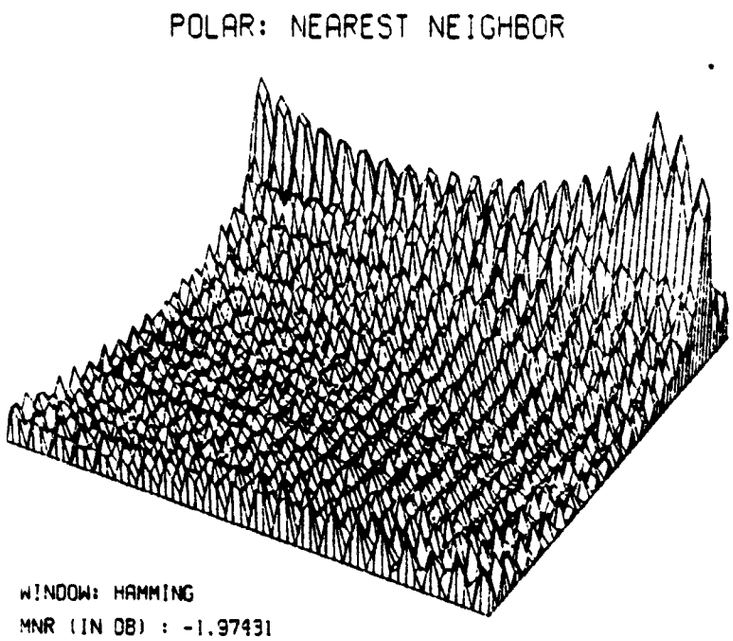


Figure 5.8 Nearest Neighbor Interpolator at (16,1).

response (see Table 5.1). This is verified by comparing the respective MNRs.

The ID and ID^2 interpolators performed only 3 dB better than NN in the (16,1) subarray (Figs. 5.13 and 5.14). Subjectively, the NN reconstruction at (16,1) (see Fig. 5.8) is extremely poor compared to ID and ID^2, and yet the MNR differ by only a few dB. Part of the reason for this lies in the fact the images are plotted with log scales where the floor is at -60 dB. This is only one-thousandth the height of the peak and so contributes negligibly to the image MNR. However, the eye views image intensity logarithmically, so the plot was scaled as such, despite the misleading MNR value.

The spurious targets are also present in the ID and ID^2 images, and there is little that can be done with these low-order algorithms to reduce their height. With this in mind, we turn to

TABLE 5.2 Evaluation Results for Fourier Piece (16,1) on Polar Grid.

Target Position	Interp.	Parameter	MNR (db)	CPU time (seconds)	|MNR/CPU|
-23,24	NN		-1.97	3.28	0.60
-23,24	G-ID	(0,1)	-4.41	19.82	0.22
-23,24	G-ID	(0,2)	-4.45	5.75	0.77
-23,24	G-ID	(0,3)	-4.30	22.57	0.19
-23,24	wsinc	2	-4.13	17.7	0.23
-23,24	wsinc	4	-7.45	28.4	0.26
-23,24	wsinc	6	-10.62	41.3	0.26
-23,24	wsinc	8	-13.76	53.0	0.26
-23,24	wsinc	10	-17.01	61.7	0.28
-23,24	wsinc	12	-20.53	70.6	0.29
-23,24	wsinc	14	-24.48	82.7	0.30
-23,24	wsinc	16	-29.10	94.7	0.31
-23,24	wsinc	18	-34.66	102.5	0.34
-23,24	wsinc	20	-41.35	112.0	0.37
-23,24	wsinc	22	-47.10	126.0	0.37
-23,24	wsinc	24	-47.91	134.0	0.36
-23,24	B-spline	-0.25	-7.67	17.22	0.45
-23,24	B-spline	-0.50	-9.29	17.22	0.54
-23,24	B-spline	-0.75	-11.06	17.22	0.64
-23,24	B-spline	-1.00	-13.07	17.22	0.76
-23,24	Spline (IMSL)		-15.35	27.7	0.55

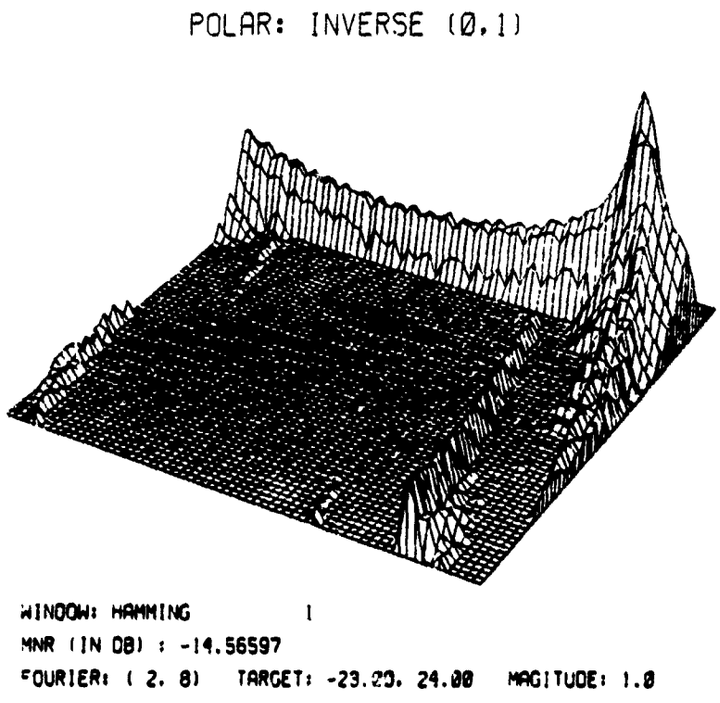


Figure 5.9 Inverse Distance Interpolator at (2,8).

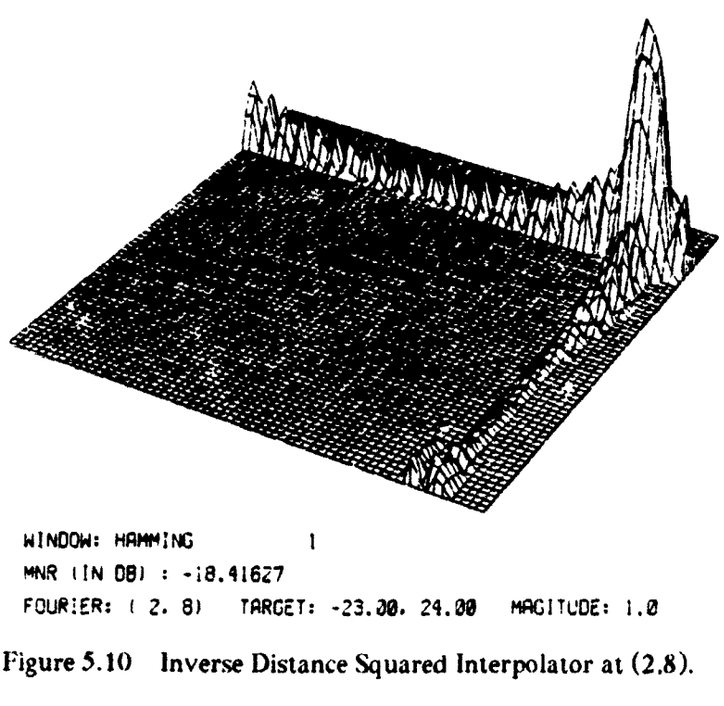


Figure 5.10 Inverse Distance Squared Interpolator at (2,8).

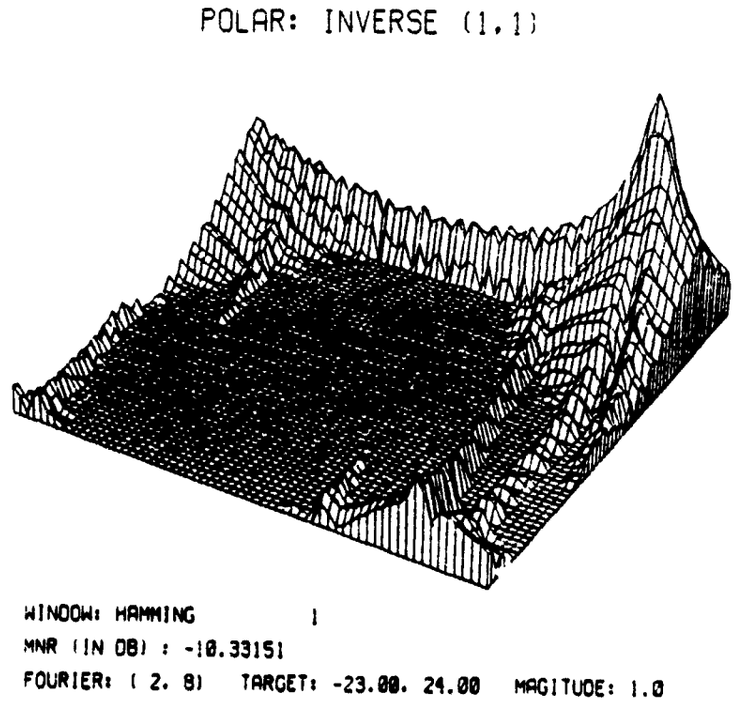


Figure 5.11 Generalized Inverse Distance Interpolator at (2,8).

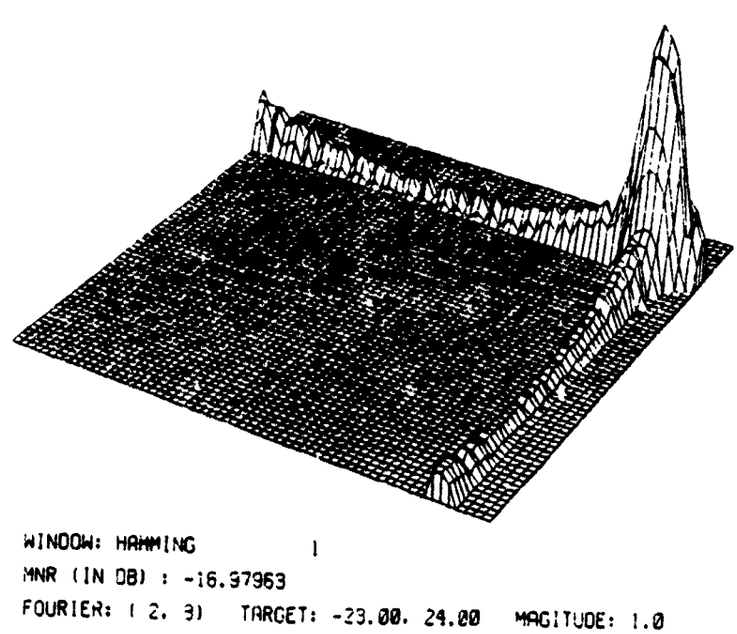


Figure 5.12 Generalized Inverse Distance Squared Interpolator at (2,8).

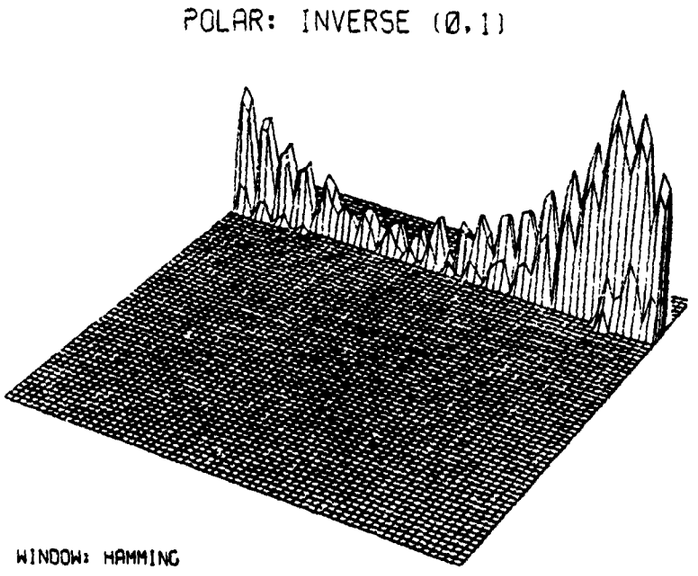


Figure 5.13 Inverse Distance Interpolator at (16,1).

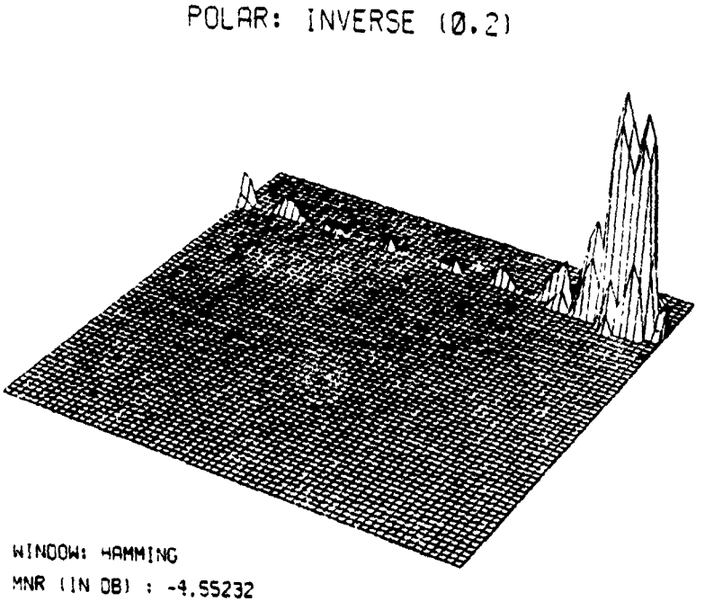


Figure 5.14 Inverse Distance Squared Interpolator at (16,1).

the more sophisticated weighted sinc interpolators.

It was shown in Chapter 3 that the sinc kernel would exactly restore a bandlimited signal which was properly (Nyquist rate, infinite length) sampled. The weighted sinc is the spatially limited approximation and the image reconstructions presented here demonstrate that even a very narrow part of the sinc performs remarkably well. Figures 5.15 and 5.16 show the results of interpolating with a windowed sinc of orders 2 and 6 in the (2,8) subarray. Even for a window of only 2 samples wide, the w-sinc produces an MNR of -18.9 dB which is much better than the ID interpolator. The CPU time is even lower for the w-sinc because it is implemented separably. If the window is widened to 6 samples, the reconstruction improves, and the MNR drops to -30.4 dB. The w-sinc size can be increased further to 10 and 14 to improve the MNR still more (Figs. 5.17 and 5.18), but soon a point is reached where the MNR fails to drop (see Table 5.2). The reconstruction is approaching the exact point target of Fig. 5.2, and so added terms to the w-sinc interpolator simply increase the algorithm cost without a noticeable performance improvement.

If the w-sinc interpolator is used in the (16,1) Fourier subarray, the order can be adjusted to reduce the spurious peak to an acceptable level. In Fig. 5.19, the w-sinc interpolator of order 6 produces an image with a secondary peak at -10.2 dB below the peak. If the order is increased to 16 (Fig. 5.20), the peak drops to -28.6. Of course, the processing time rises proportionately from 41.3 seconds to 94.7 seconds, the ratio of which, 0.44, is roughly 6/16 (0.38), i.e., the algorithm processing time is approximately linear with interpolator order. If the order is increased to 20, the secondary peak disappears from the image (it is below 60 dB).

The processing time of the NN and GID interpolators is relatively small compared to the w-sinc of any order greater than 2. The strength of NN and GID lies in speed, though, rather than high quality image reconstruction. The processing times for each of the various interpolators are plotted against interpolator order in Fig. 5.21. The NN algorithm is referred to as zeroth order, and the GID order refers to the power of the inverse distance, i.e., ID is first order

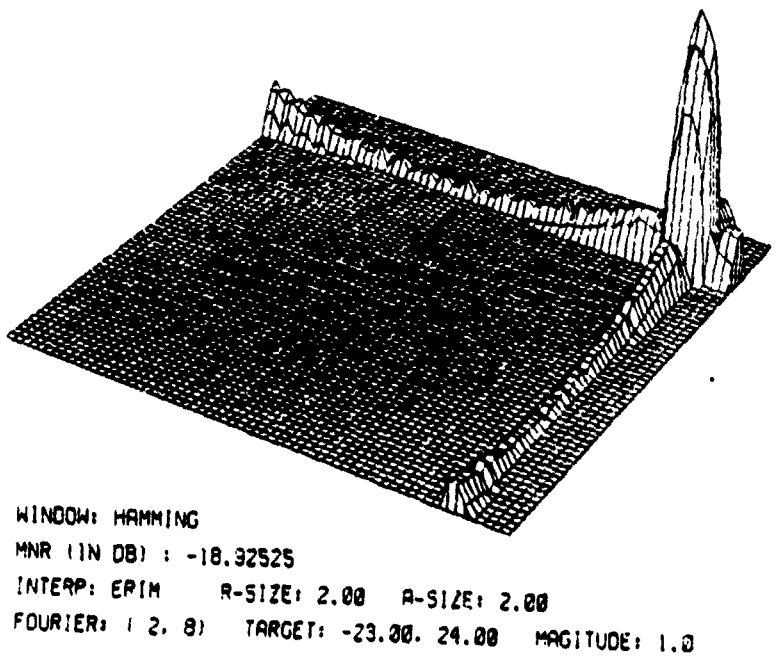


Figure 5.15 W-Sinc Interpolator of order 2 at (2,8).

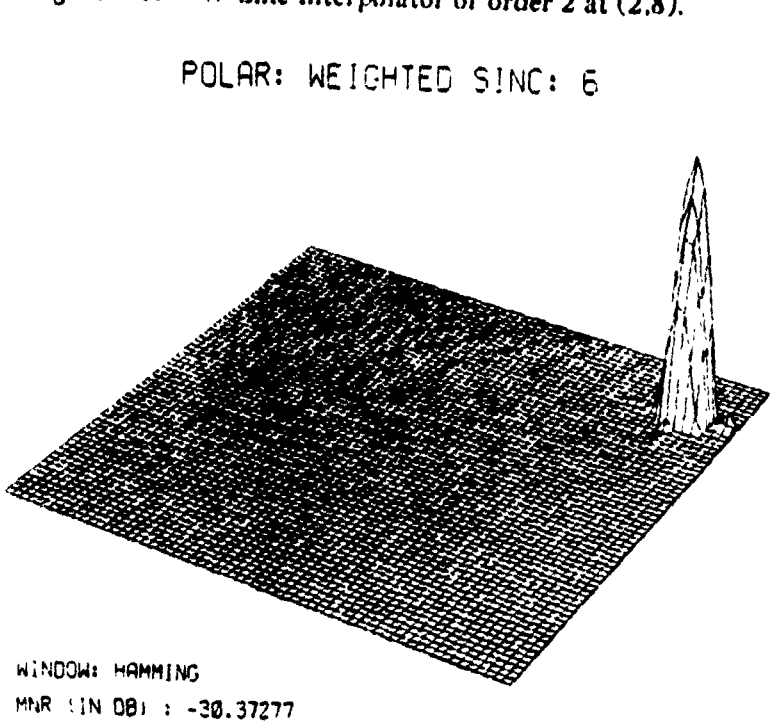


Figure 5.16 W-Sinc Interpolator of order 6 at (2,8).

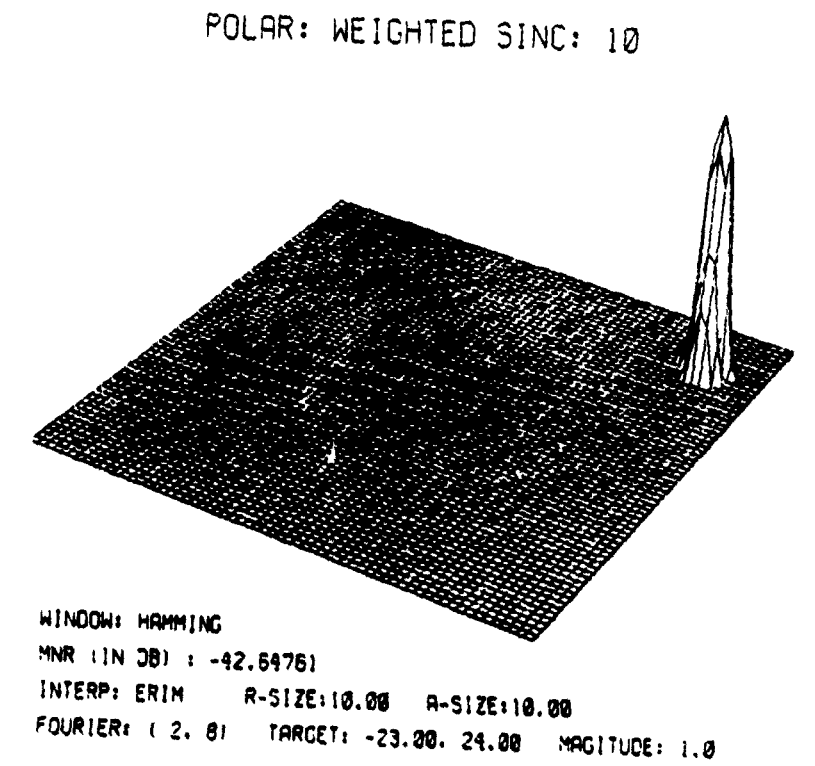


Figure 5.17 W-Sinc Interpolator of order 10 at (2,8).

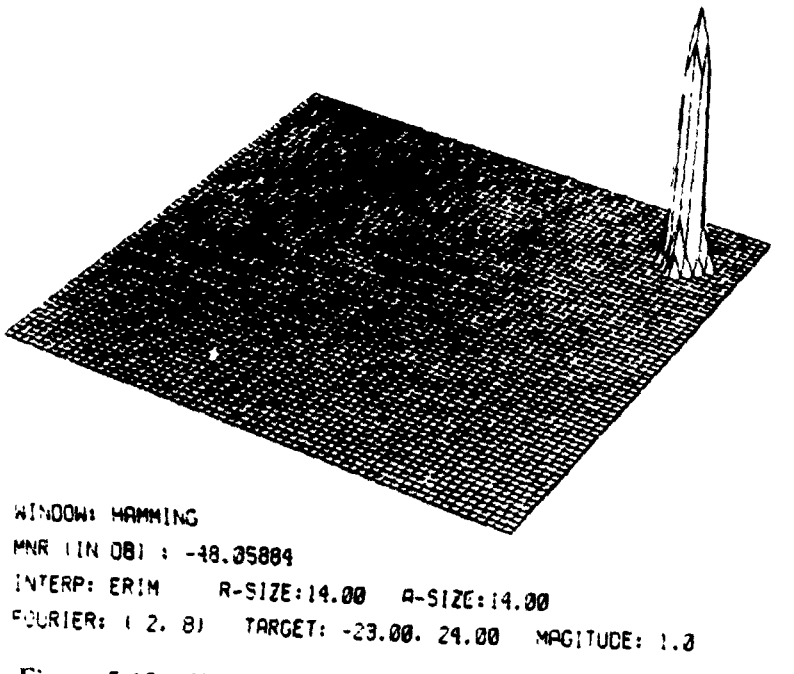


Figure 5.18 W-Sinc Interpolator of order 14 at (2,8).

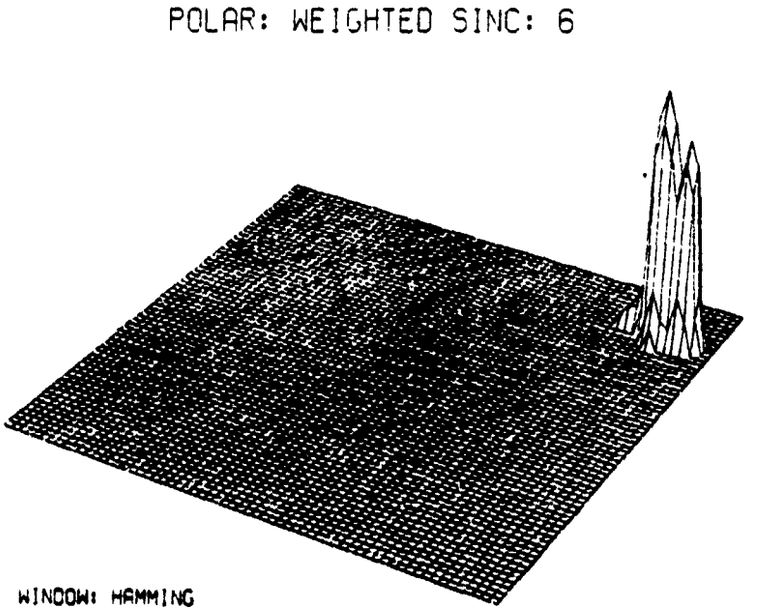


Figure 5.19 W-Sinc Interpolator of order 6 at (16.1).

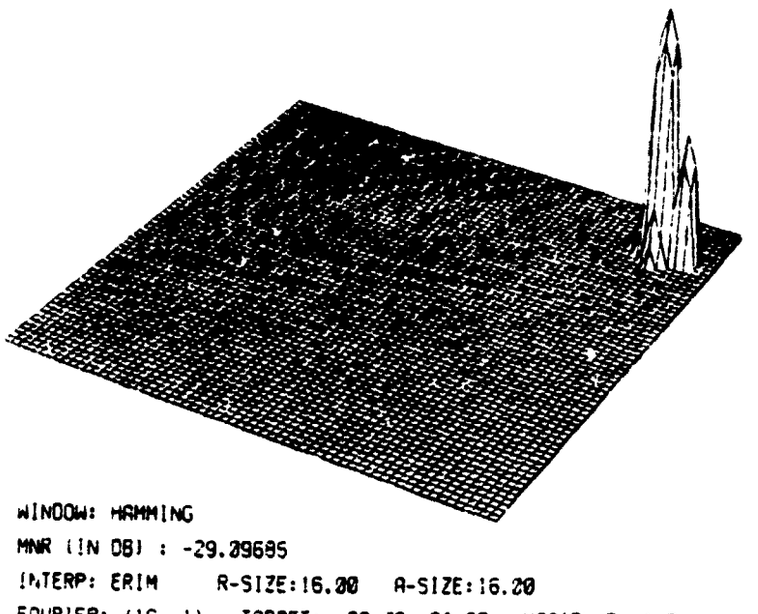


Figure 5.20 W-Sinc Interpolator of order 16 at (16.1).

and ID^2 is second order.

The IMNRI/CPU ratio is plotted in Fig. 5.22 for the w-sinc interpolator as applied to several of the subarrays. Several observations can be made by studying this family of curves. First, the w-sinc curves all converge to the value 0.36 dB/s. We would expect convergence to this ratio since each w-sinc can become arbitrarily close to the exact response MNR, and the processing time of w-sinc is independent of the subarray. Notice that the ordering of the curves from top to bottom corresponds to the increasing value of the range subscript of the subarray coordinate pair. This indicates that the more distant subarrays require higher order w-sinc interpolators to achieve the same MNR values. This is due to the increasing sample spacing which generates the spurious targets. These targets can only be removed by increasing interpolator order at the expense of processing time.

The IMNRI/CPU value can be thought of as a benefit/cost ratio. A higher ratio indicates a better reconstruction for CPU resources used. The slope indicates the amount of *improved* MNR for the amount of *additional* CPU time used. Since CPU time is proportional to interpolator order, a more negative slope corresponds to a decrease in the amount of MNR improvement. Ultimately, this means that the size of the interpolator can be varied as it is moved out in the radial direction to achieve the same MNR for each subarray. This concept is discussed in Chapter 6 under Further Research.

The last algorithm shown here for the polar grid is the cubic spline and B-spline interpolators described in the last section of Chapter 3. The first version of the splines, the complete cubic spline, is implemented much like the w-sinc algorithm. The data are first interpolated to a keystone grid by calculating the spline coefficients for each interval and then evaluating the corresponding cubic polynomial at the intermediate (keystone) points. The bulk of the processing is consumed in calculating the coefficients via the matrix solution step. Evaluation of the cubic polynomial is relatively fast compared to the calculation of the polynomial coefficients. Next, a cubic spline is generated in the azimuth direction and evaluated along the rectangular

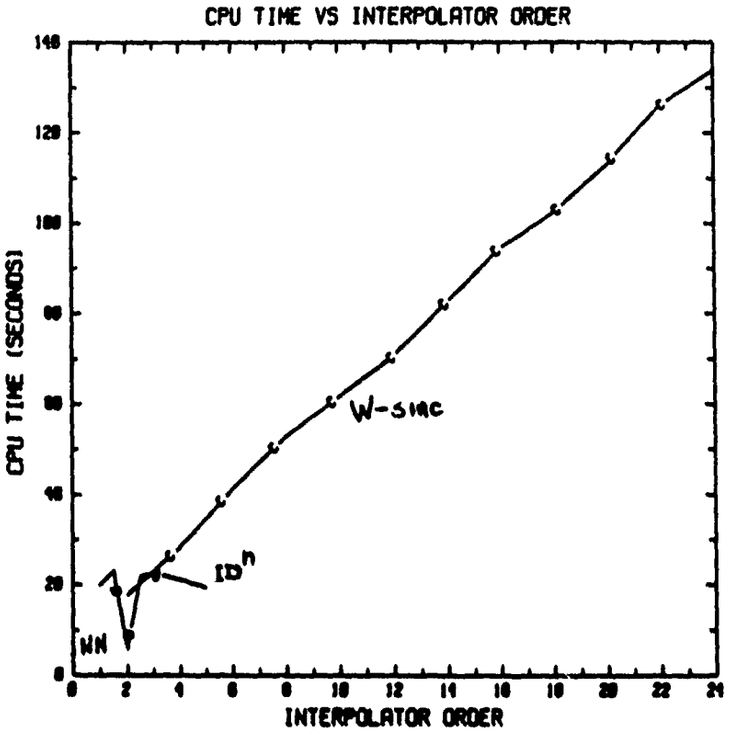


Figure 5.21 CPU Time Versus Interpolator Order at (2,8).

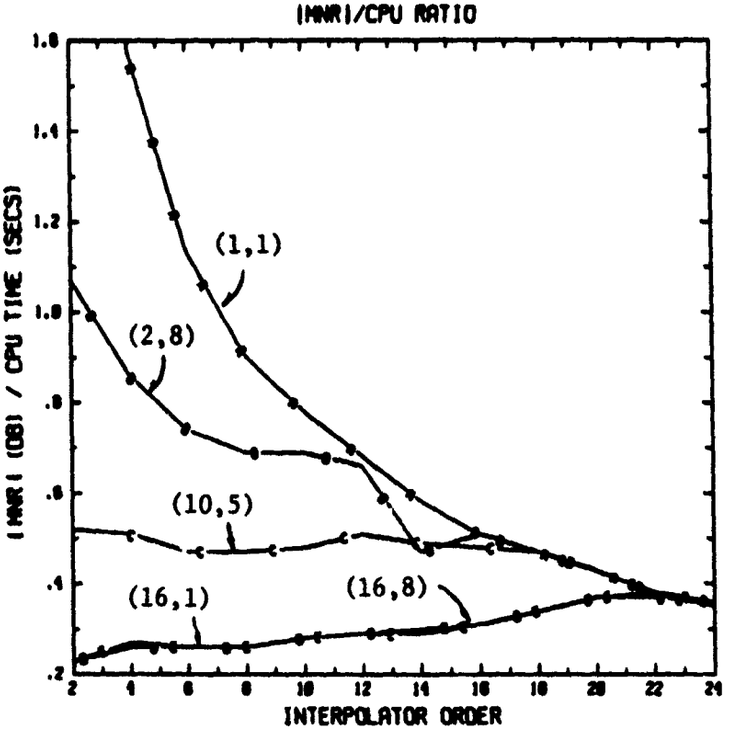


Figure 5.22 IMNRI/CPU ratio for W-sinc in Different Subarrays.

grid coordinates. As shown in Table 5.1 for the subpatch (2,8), the MNR is roughly that of a windowed sinc of an order between 6 and 8 (the interpolator order need not be even, nor must it be an integer), yet processing time is close to the w-sinc of order 4. The cubic spline reconstruction is shown in Fig. 5.23. The IMNRI/CPU ratio is much higher for the cubic spline than it is for the w-sinc, since it performs so much better for the equivalent processing cost.

The modified B-splines also do quite well compared to the w-sinc interpolator. They were implemented the same way as the w-sinc, i.e., with the intermediate keystone grid. Four different values of the parameter in Eq. (3.52) are used, -0.25, -0.50, -0.75, and -1.00, with -1.0 providing the best reconstruction. The MNR of -30.46 is only 3 dB worse than the complete cubic spline, and the processing time is much lower (driving the IMNRI/CPU ratio up). It is shown in Fig. 5.24.

Chapter 4 described several windows which may be applicable for SAR data. The effects of different windows may be seen by examining the image reconstructions from a typical interpolator: for example, a 10th order w-sinc. If a uniform window is applied, i.e., no weighting function, then the reconstruction is a very narrow spike surrounded by some low sidelobes (Fig. 5.25a). This is a result of proper target placement so that the output sinc is sampled in the nulls. The sidelobes are a result of interpolator error. If the Fourier data is windowed with a disk shaped 1-0 weighting function, then the MNR is dramatically worsened to -4.87 dB and the image looks very much like an NN interpolation (Fig. 5.25b).

The separable Hamming window, the standard used in the evaluations, has an image reconstruction of the same data shown in Fig. 5.26a. The MNR is a very low -42.65 dB. The output of a circular Hamming window is much noisier in appearance (Fig. 26b) and has an MNR of -28.12 dB. It is apparent that the discontinuity at the Hamming edge is causing some problems. There is an additional, more subtle effect occurring as a result of using the circular Hamming window. The zeros of the circular Hamming Fourier transform do not fall on the Cartesian grid sample points. Thus, even an exact target reconstruction will display sidelobes

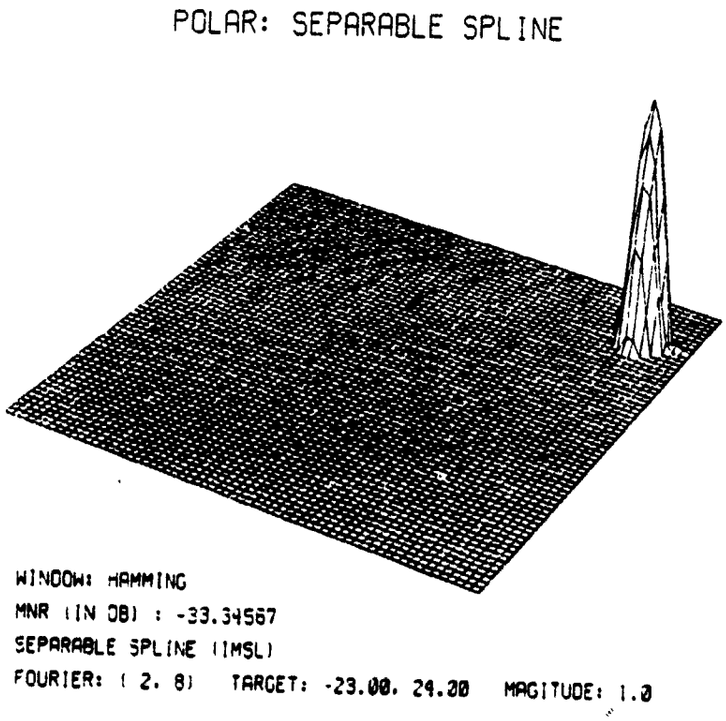


Figure 5.23 Complete Cubic Spline Interpolator at (2,8).

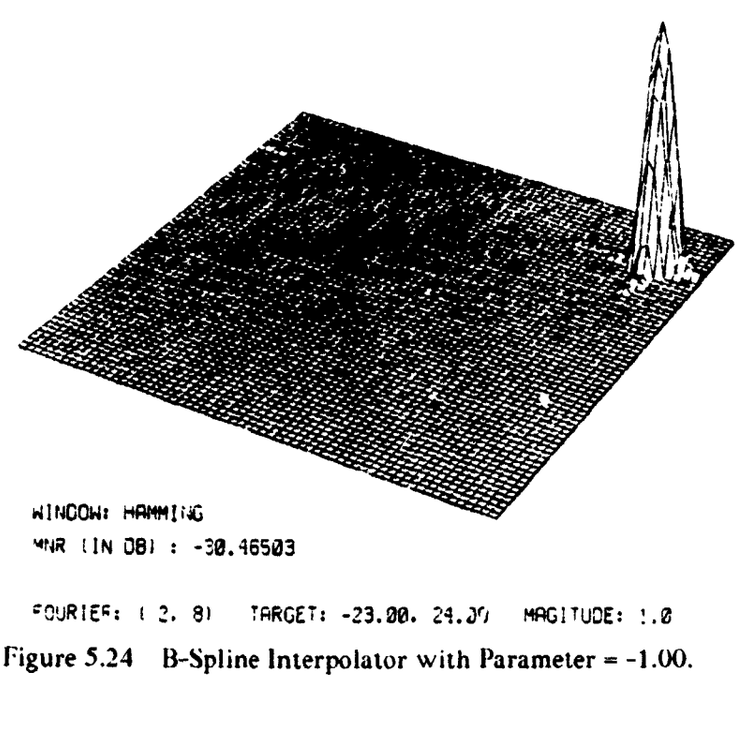


Figure 5.24 B-Spline Interpolator with Parameter = -1.00.

which could be misleading. This can be corrected slightly through the use of the Hanning window which goes to zero smoothly. The Hanning window causes faster sidelobe decay, while having a slightly higher first sidelobe than the Hamming. Figures 5.27a and 5.27b display the results of the Hanning window, both separable and circular. Note that the separable Hanning window comes very close to matching the separable Hamming window with an MNR of -40.59 dB.

5.5 Equi-PRF Grid

The equi-PRF grid gained little in the way of performance. Figures 5.28 and 5.29 are examples of the nearest neighbor algorithm applied to the equi-PRF grid in the regions (2,8) and (16,1) which may be compared to Figs. 5.5 and 5.8 for the standard polar format. (Tables C.5 to C.18 show processing time and MNR values for each of the equi-PRF algorithms.) The processing time was slightly improved for the w-sinc algorithm because the azimuth pass (interpolation on the intermediate keystone grid) calculations were simplified. With the standard polar grid, the keystone samples were unevenly spaced in azimuth, resulting in excessive coordinate positional information involving trigonometric functions. With equally spaced data, positional information is calculated as a simple real multiply. For splines and B-splines, the differences were very slight, corresponding to the time taken to determine into which interval (between which two knots) the output point falls prior to polynomial evaluation. With non-uniform spacing in azimuth, a binary search is performed to determine the interval, but with equally spaced data, the interval is again found with one real multiply.

5.6 Keystone Grid

Since the keystone grid provides data along vertical lines, the interpolation procedure need act only in one dimension. It is much like *beginning* with the second stage of the w-sinc output, but from an exact first stage interpolation. Thus, a better reconstruction should be

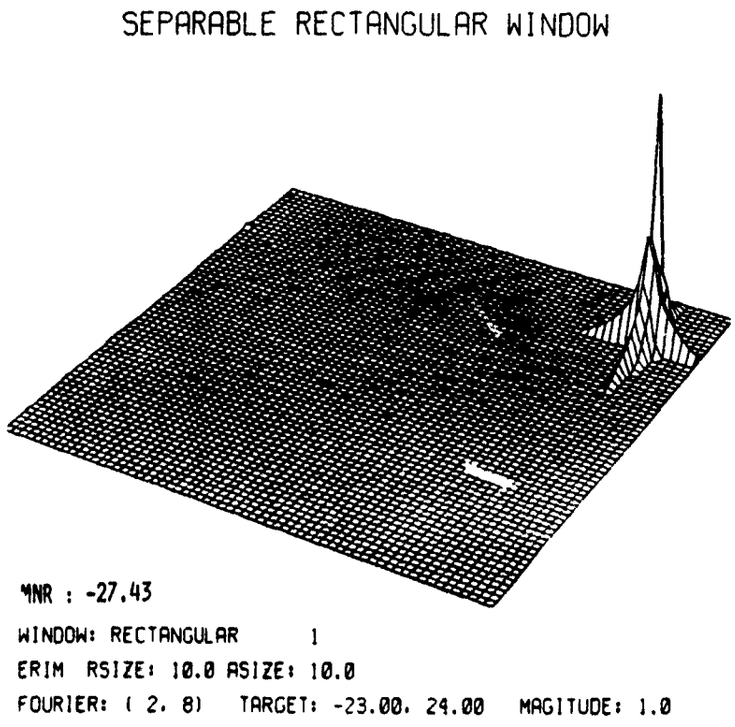


Figure 5.25a Uniform Window After 10th Order W-Sinc Interpolator.

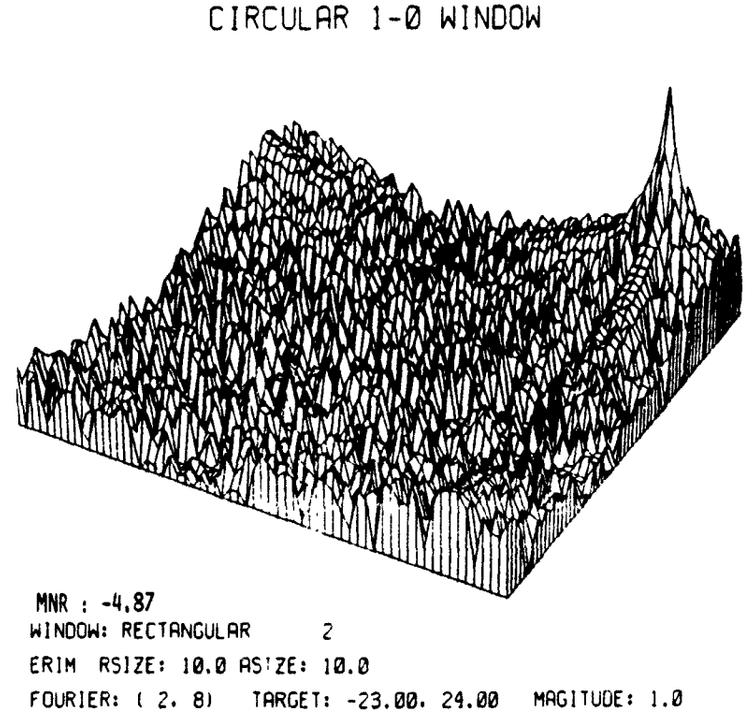


Figure 5.25b Circularly Symmetric Uniform Window (a Disk).

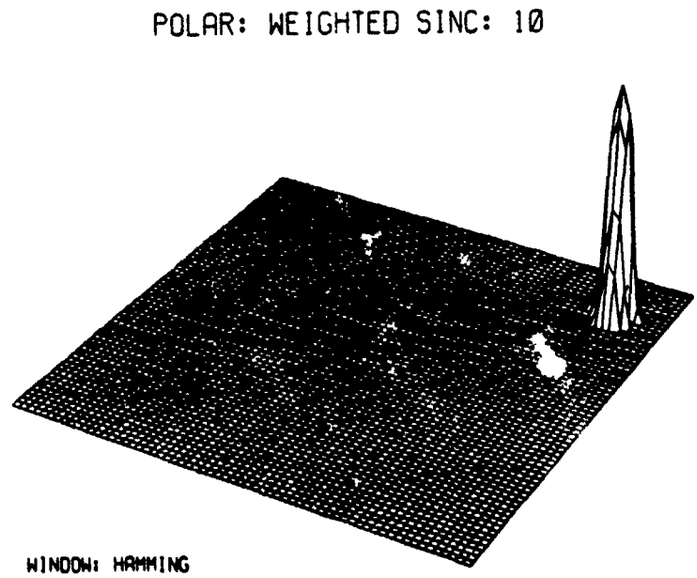


Figure 5.26a Interpolated Data With Separable Hamming Window.

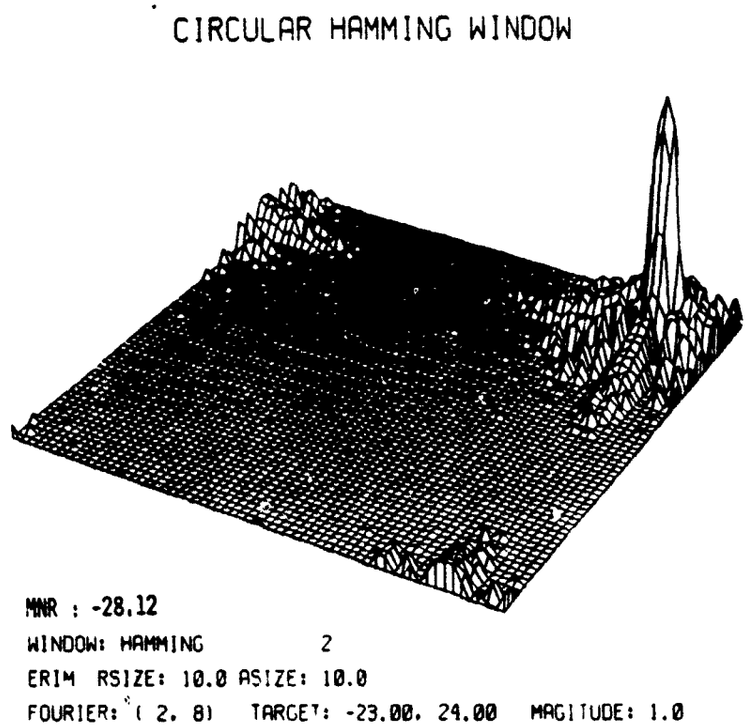
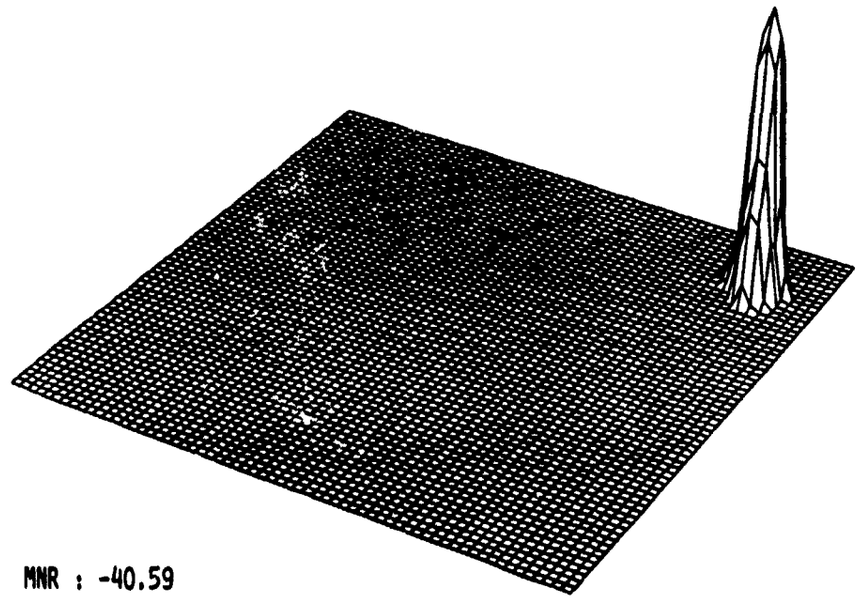


Figure 5.26b Interpolated Data With Circular Hamming Window.

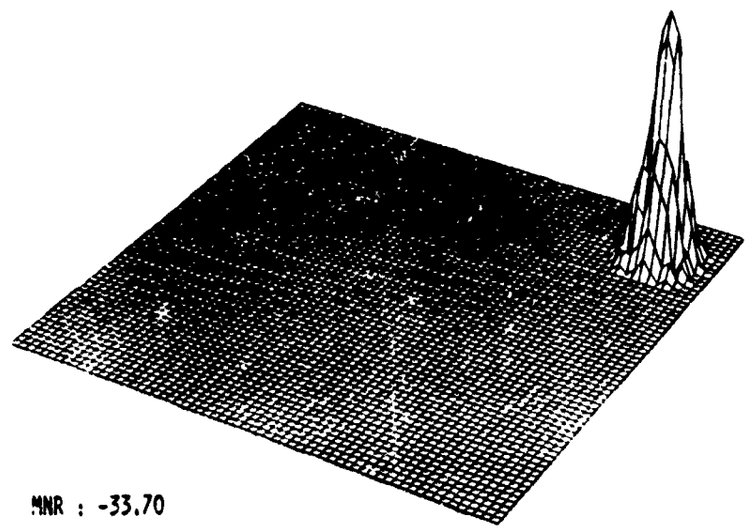
SEPARABLE HANNING WINDOW



MNR : -40.59
WINDOW: HANNING 1
ERIM RSIZE: 10.0 ASIZE: 10.0
FOURIER: (2. 8) TARGET: -23.00. 24.00 MAGITUDE: 1.0

Figure 5.27a Interpolated Data With Separable Hanning Window.

CIRCULAR HANNING WINDOW



MNR : -33.70
WINDOW: HANNING 2
ERIM RSIZE: 10.0 ASIZE: 10.0
FOURIER: (2. 8) TARGET: -23.00. 24.00 MAGITUDE: 1.0

Figure 5.27b Interpolated Data With Circular Hanning Window.

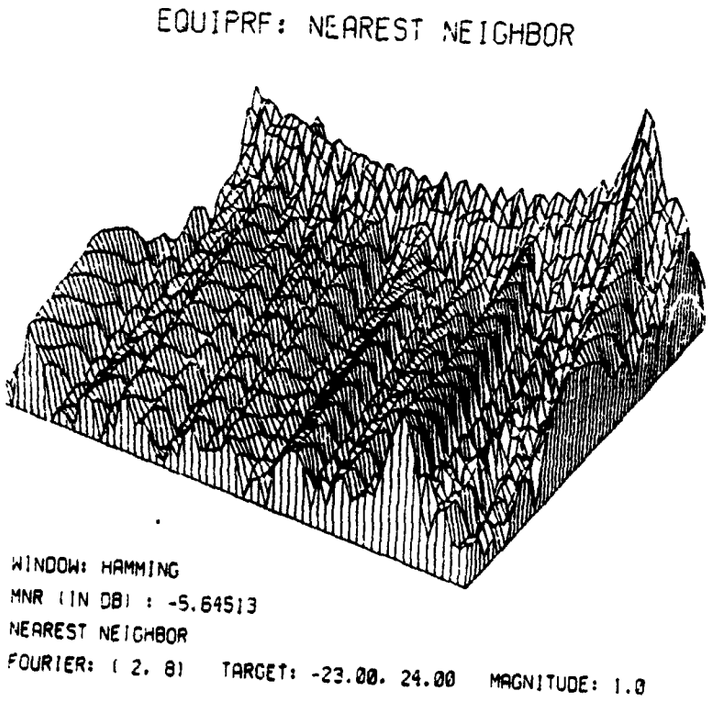


Figure 5.28 Equi-prf Grid with Nearest Neighbor Interpolator (2,8).

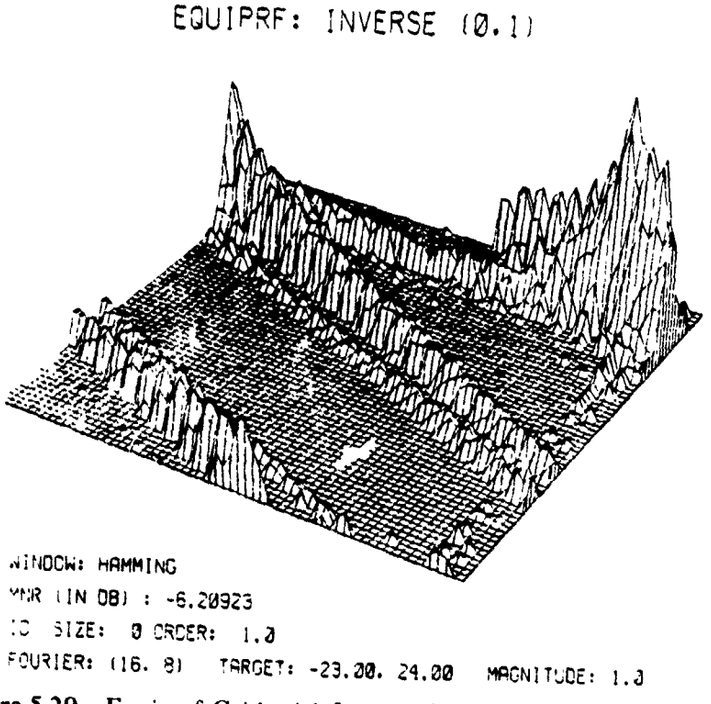


Figure 5.29 Fqui-prf Grid with Inverse Distance Interpolator (16,1).

observed (Figs. 5.30 through 5.41). The results substantiate this theory as seen in Tables C.9 to C.12. Every MNR value is noticeably better with the keystone data format than for the polar or equi-PRF format. It is interesting to note that the w-sinc interpolator MNR values for the keystone grid converge much faster to a minimum than the polar format data. This is particularly true for the (16,1) subarray where the spurious targets account for poor MNRs. Elimination of the range line interpolation step has removed the error introduced into the azimuthal interpolation stage. In the polar data format, this error could only be minimized by increasing interpolator order, and thus processing time, to obtain the same MNR as the keystone data format. The interpolators also run much faster because the spatial position calculations are significantly reduced (only one dimensional calculations) and the two-stage algorithms now only operate in one stage.

The keystone geometry permits the use of the chirp-z algorithm along the azimuth data lines as described in Chapter 4. This combines both interpolator and FFT sections into one stage. Since the available input array was 94 by 94 samples, the azimuth input vector size was 94 with an output vector size of 64 (in the spatial domain). The results of the chirp-z algorithm are shown in Figs. 5.35 and 5.41 for subarrays (2,8) and (16,1), respectively. The MNR values of -1.32 dB for (2,8) is surprisingly poor compared to the other keystone interpolators, while its performance in the (16,1) region was similar to a w-sinc interpolator with order 12 (in speed and MNR). The processing time is high because the FFT included in the algorithm must be zero padded to perform a non-cyclic convolution. This padding increases the FFT size from 64 to 256 (for 94 input points, 256 is the first power of two greater than 94+64).

Note in Fig. 5.41 that the spurious side lobes are significantly lower than the w-sinc order 16 for the same subarray. This is because the chirp-z algorithm interpolates/transforms in such a manner that the spurious target analysis of Chapter 4 is not applicable. The high spurious target seen in Figs. 5.19 and 5.20 disappears in 5.41, because the sampling rate is not changed in the Fourier domain, but rather, the transform is calculated directly from the input

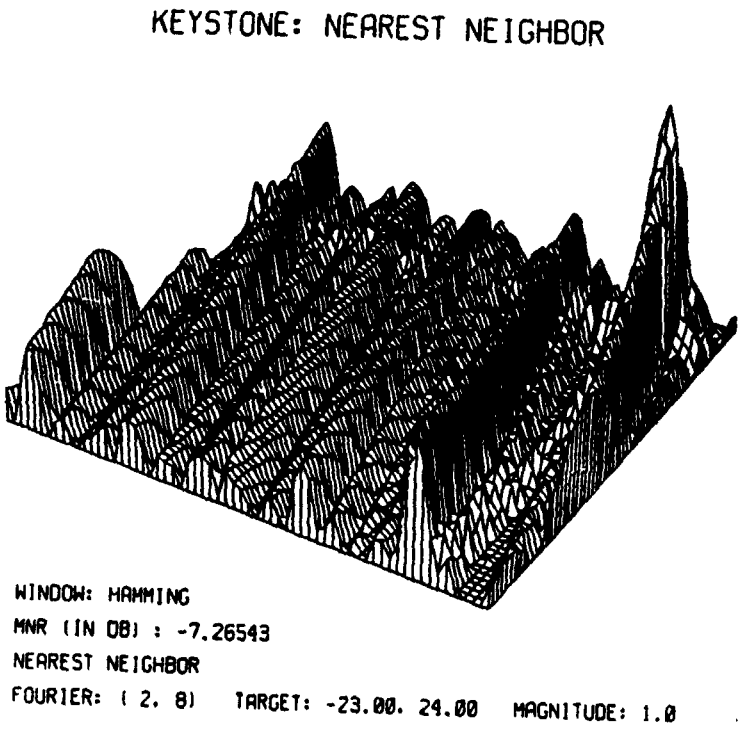


Figure 5.30 Keystone Grid with Nearest Neighbor Interpolator at (2,8).

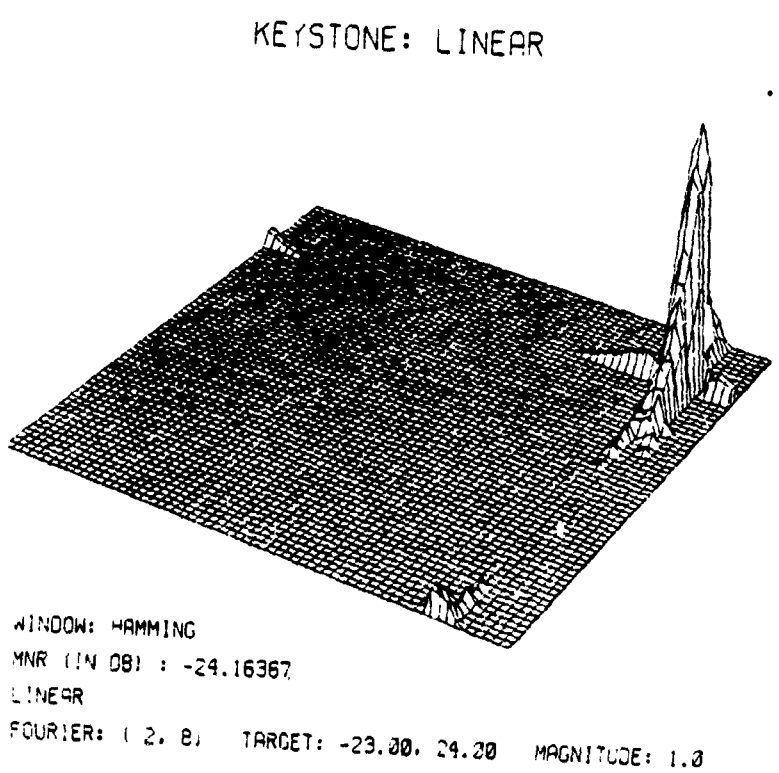


Figure 5.31 Keystone Grid with Linear Interpolator at (2,8).

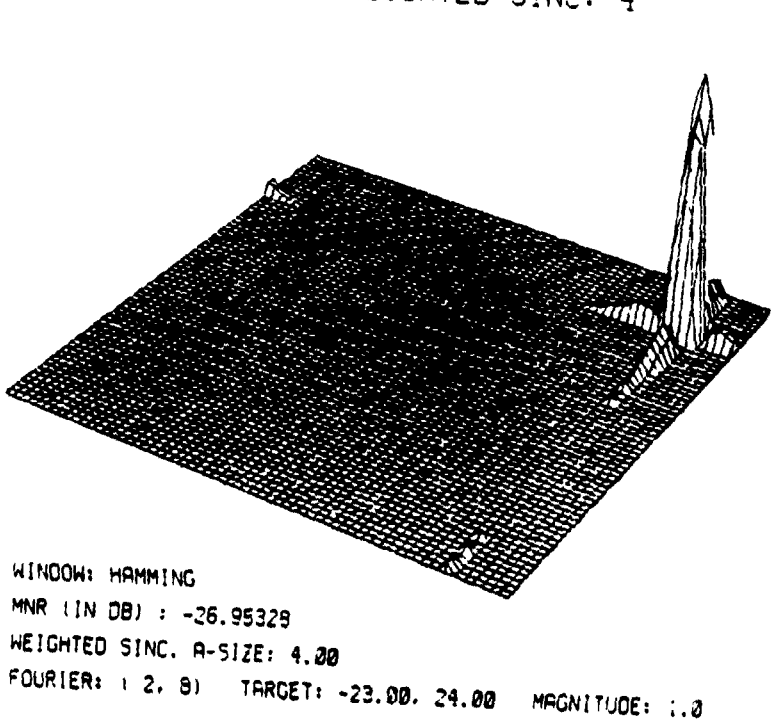


Figure 5.32 Keystone Grid with W-Sinc Order 4 at (2,8).

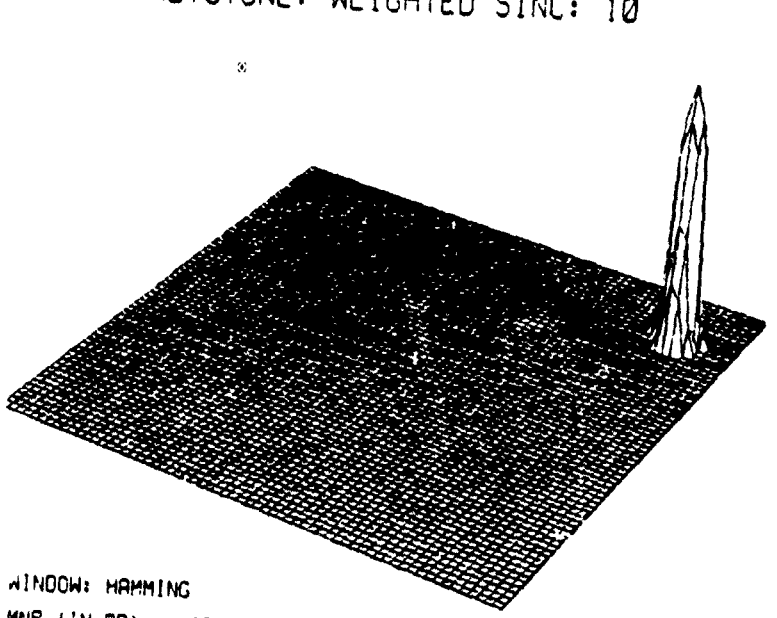


Figure 5.33 Keystone Grid with W-Sinc Order 10 at (2,8).

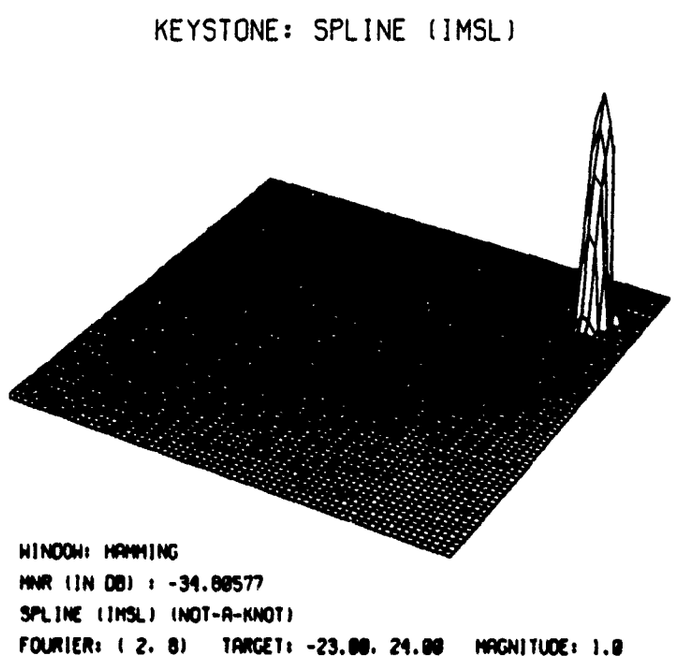


Figure 5.34 Keystone Grid with Cubic Spline Interpolation at (2,8).

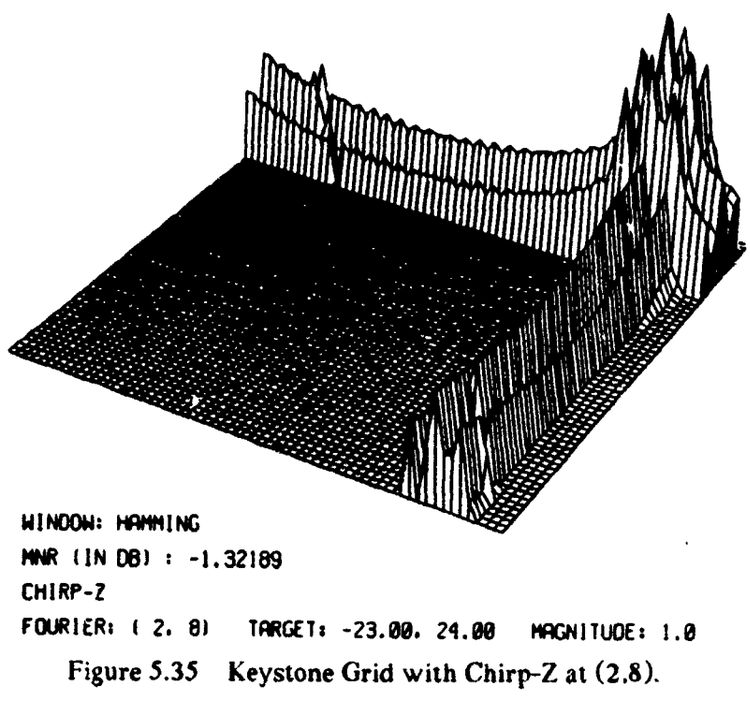


Figure 5.35 Keystone Grid with Chirp-Z at (2,8).

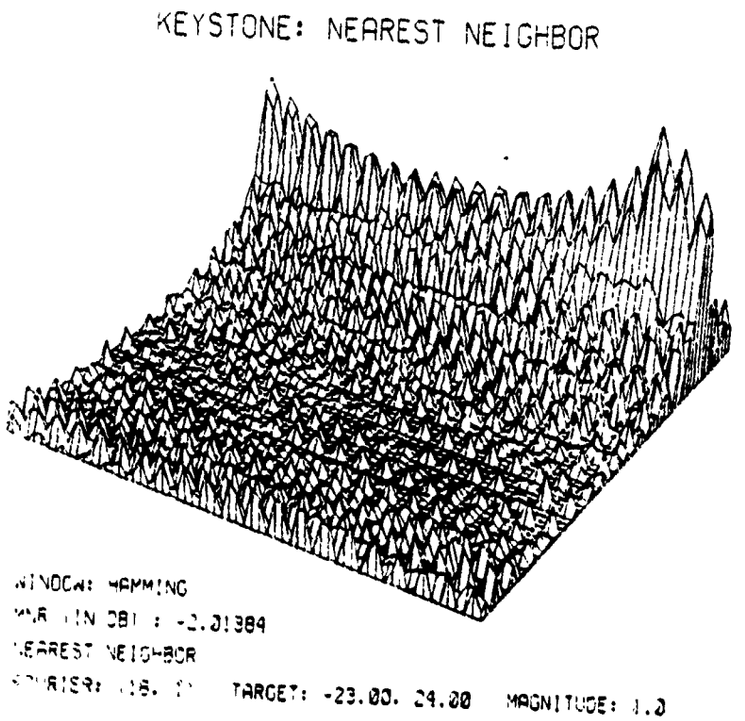


Figure 5.36 Keystone Grid with Nearest Neighbor at (16,1).

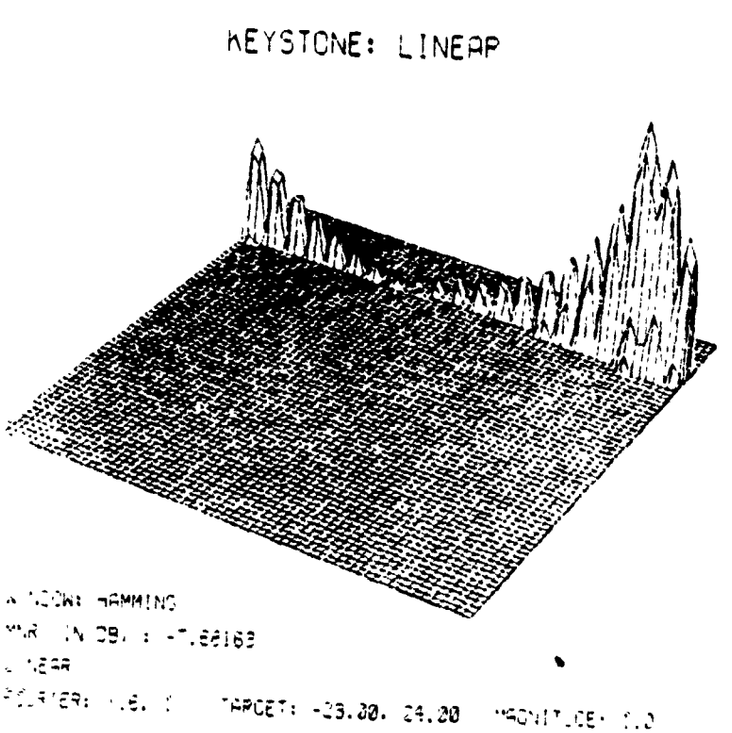


Figure 5.37 Keystone Grid with Linear Interpolator at (16,1).

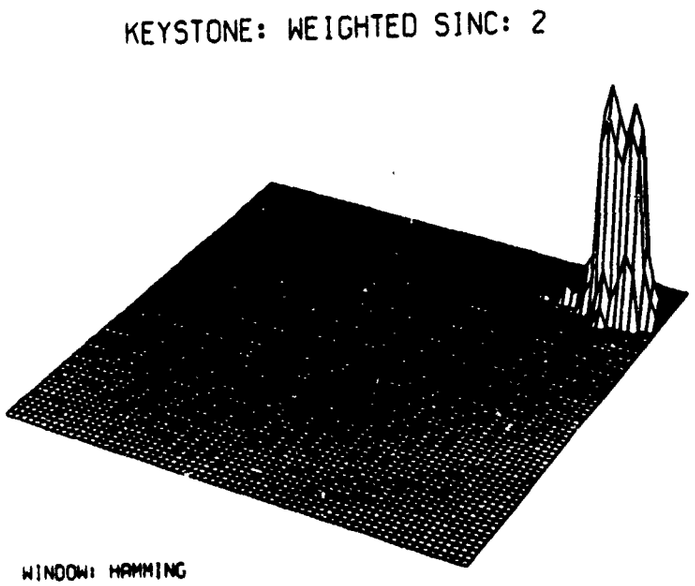


Figure 5.38 Keystone Grid with W-Sinc Order 2 at (16,1).

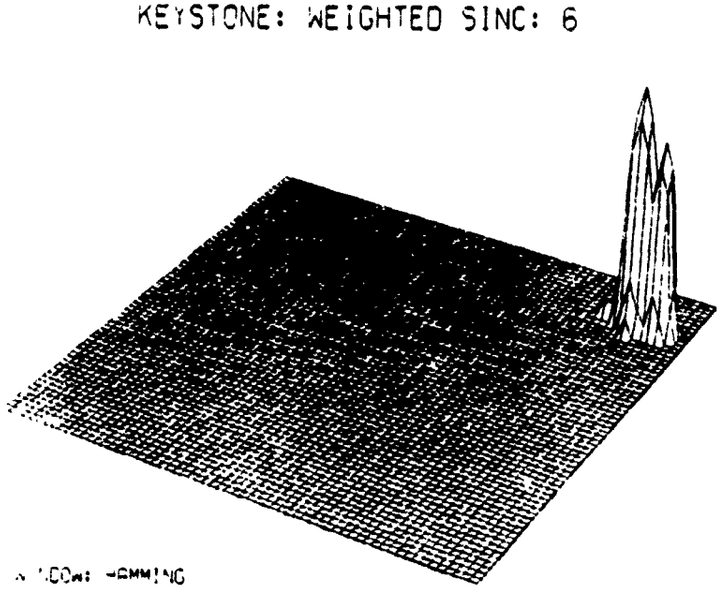


Figure 5.39 Keystone Grid with W-Sinc Order 6 at (16,1).

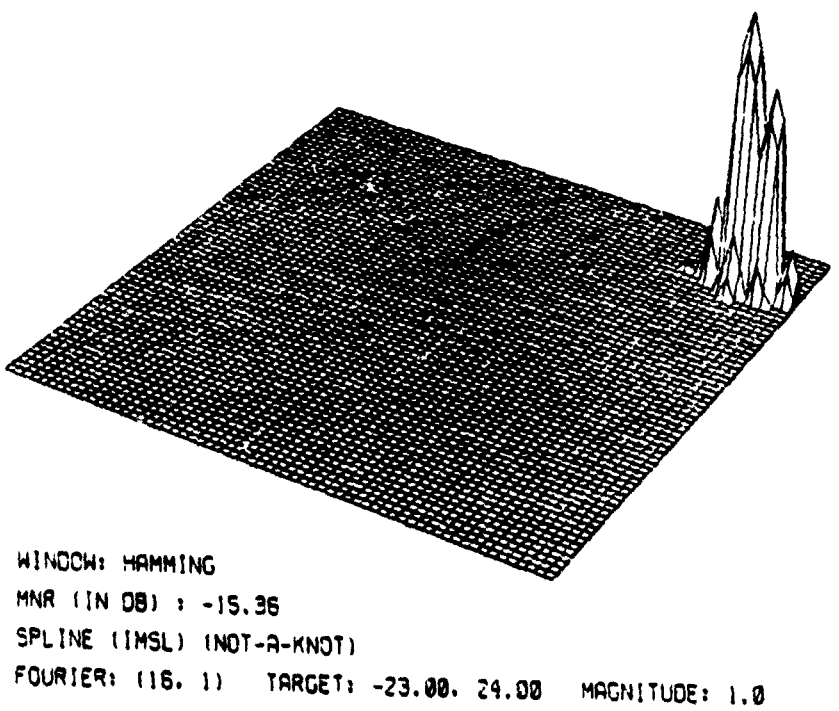


Figure 5.40 Keystone Grid with Cubic Spline Interpolation at (16,1).

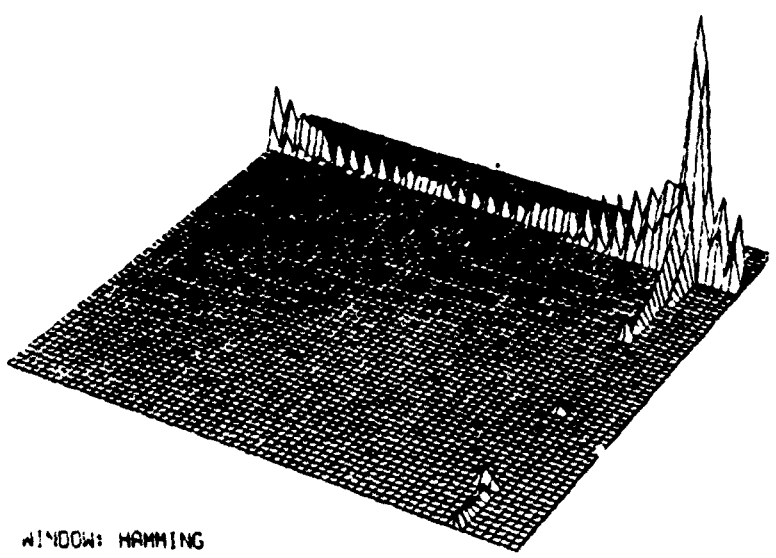


Figure 5.41 Keystone Grid with Chirp-Z at (16,1).

data. Theoretically, the chirp-z transform should behave as an ideal interpolator (for a limited data record).

5.7 Complexity Analysis

Using the methodology for complexity analysis discussed above, an expression for each of the interpolators was calculated. This was done for the polar and keystone grids. Complexity measures for the equi-prf grid are similar to the polar format, but are minus the trigonometry terms which are required for the data position calculations in the second stage of the separable interpolators. For each algorithm, a general expression is presented which contains a break-down of the software in terms of the basic cost units ($C_{a/s}$, $C_{m/d}$, C_{trig}, etc.). This is then reduced to an intermediate expression by making certain assumptions about the input array size (number of input azimuth lines and $N^2 >> N$). Finally, a simplified expression is derived by assuming properties of the data processor (computation times for a C_{trig}, $C_{a/s}$, $C_{m/d}$, etc.). It should be kept in mind that this simplified expression is only as accurate as the assumptions made in the approximation. Recall that the FFT complexity is part of these expressions. It can be identified as the $2N\log_2 N$ term in the expressions. The variables N, I_r, I_a, and K correspond to the output grid size (N by N), the number of input range samples, the number of input azimuth samples and the interpolator order (w-sinc only), respectively. Comparisons of the complexity measures are done with N=1024 corresponding to the original image size. The nearest neighbor complexity is given by (5.1a) through (5.1c):

General expression for nearest neighbor complexity:

$$C_{NN} = (5N^2 + N + 2N^2\log_2 N)C_{m/d} + (5N^2 + N + 2N^2\log_2 N)C_{a/s} + N^2 C_{trig} + N^2 C_{sqrt} \qquad (5.1a)$$

Intermediate expression ($N^2 >> N$):

$$C_{NN} \approx (5 + 2Log2N)N^2C_{m/d} + (5 + 2\log_2N)N^2\,C_{a/s} \quad (5.1b)$$
$$+ N^2C_{trig} + N^2C_{sqrt}$$

Simplified expression ($C_{a/s}$=C_{trig}=C_{sqrt}=C, $C_{m/d}$=5C):

$$C_{NN} \approx (32 + 12\log_2N)N^2\,C \quad (5.1c)$$

Equation (5.1c) shows that despite the simplicity of the nearest neighbor algorithm, it is still relatively expensive compared to the FFT times. This is primarily due to the square root and trigonometric calculations required to locate the rectangular output points in polar space.

Since the ID^2 interpolator outperformed ID in both speed and MNR, it was the only ID type of algorithm analyzed. It is given in 5.2a through 5.2c.

General expression for Inverse Distance squared (ID^2) complexity:

$$C_{ID^2} = (32 + 2\log_2N)\,N^2\,C_{m/d} + (29 + 2\log_2N)N^2C_{a/s} + N^2C_{trig} \quad (5.2a)$$

Intermediate expression for ID^2 ($I_a = I_r = N$)

$$C_{ID}^2 \approx (32 + 2\log_2N)N^2C_{m/d} + (29 + 2\log_2N)N^2C_{a/s} + N^2C_{trig} \quad (5.2b)$$

Simplified expression for ID^2 ($C_{a/s}$=C_{trig}=C_{sqrt}=C, $C_{m/d}$=5C)

$$C_{ID}^2 \approx (190 + 12\log_2N)N^2\,C \quad (5.2c)$$

General expression Weighed Sinc Interpolator of Order K:

$$C_{Wsinc} = (4I_a + 4NI_a + 3KNI_a + N + 2NI_a + 3N^2 + 2K\,N^2 + 2N^2\log_2N)C_{m/d} \quad (5.3a)$$
$$+ (4I_a + 5NI_a + 2KNI_a + N + NI_a + 6N^2 + 3KN^2 + 2N^2\log_2N)C_{a/s}$$
$$+ (I_a + NI_a + 2N^2)C_{trig} + (KNI_a + KN^2)C_{sinc}$$

Intermediate expression with $I_a = N$

$$C_{Wsinc} \approx (9 + 2\log_2N + 5K)\,N^2\,C_{m/d} \quad (5.3b)$$
$$+ (12 + 2\log_2N + 5K)\,N^2\,C_{a/s}$$

$$+ (3)\, N^2\, C_{trig}$$

$$+ 2K\, N^2\, C_{sinc}$$

Simplified expression ($C_{a/s}=C_{trig}=C_{sqrt}=C$, $C_{m/d}=5C$)

$$C_{Wsinc} \approx (62 + 12\log_2 N + 30K)\, N^2\, C \qquad (5.3c)$$

General expression for Cubic Spline Interpolator:

$$C_{spline} = (4I_a + 10NI_a + 10NI_a + N + 8NI_a + 6N^2 + 14\, N^2 + 2N^2\log_2 N)C_{m/d} \qquad (5.4a)$$

$$+ (4I_a + 10NI_a + 16NI_a + 2N + 4NI_a + 10N^2 + 20N^2 + 2N^2\log_2 N)C_{a/s} + (I_a + N)\, C_{trig}$$

Intermediate expression:

$$C_{spline} \approx (48 + 2\log_2 N)\, N^2\, C_{m/d} \qquad (5.4b)$$

$$+ (60 + 2\log 2N)\, N^2\, C_{a/s}$$

$$+ 2N\, C_{trig}$$

Simplified expression:

$$C_{spline} \approx (300 + 12\log_2 N)\, N^2\, C \qquad (5.4c)$$

The cubic spline interpolator has a very high order of complexity due to the matrix solution step. It has an interpolator complexity which approximates that of the w-sinc of order 8. Tables C.1 through C.4 show that the performance is similar to the w-sinc with an order between 6 and 8.

General expression for B-spline Interpolator:

$$C_{B-spline} = (4I_a + 4NI_a + 12NI_a + N + 2NI_a + 3N^2 + 8\, N^2 + 2N^2\log_2 N)C_{m/d} \qquad (5.5a)$$

$$+ (4I_a + 5NI_a + 8NI_a + N + NI_a + 6N^2 + 12N^2 + 2N^2\log_2 N)C_{a/s} + (I_a + NI_a)\, C_{trig}$$

Intermediate expression:

$$C_{B-spline} \approx (29 + 2\log_2 N)\, N^2\, C_{m/d} \qquad (5.5b)$$

$$+ (30 + 2\log 2N)\, N^2\, C_{a/s}$$

$$+ N^2\, C_{trig}$$

Simplified expression:

$$C_{B\text{-}spline} \approx (176 + 12\log_2 N)\, N^2\, C \tag{5.5c}$$

The B-splines have a complexity which is approximately half that of the complete cubic spline, yet their performance is comparable. The complexity also approximates the w-sinc of order 4. This is expected, since the fourth order w-sinc and B-spline interpolators operate the same way: by convolving with 4 input points in 2 stages.

The keystone grid reduced the complexity measure by roughly a factor of 2. This is due to the elimination of one stage of the separable interpolators and the 2D positional calculations for the lower order algorithms. Equations (5.6) through (5.8) present the complexity expressions for the interpolators used on the keystone grid.

General expression for complexity of NN on keystone grid:

$$C_{NN}^{key} = (3N^2 + 2N^2\log_2 N + N)C_{m/d} + (3N^2 + 2N^2\log_2 N + N)C_{a/s} + n2C_{trig} \tag{5.6a}$$

Intermediate expression:

$$C_{NN}^{key} \approx (3 + 2\log_2 N)N^2 C_{m/d} + (3 + 2\log_2 N)N^2 C_{a/s} + N^2 C_{trig} \tag{5.6b}$$

Simplified expression:

$$C_{NN}^{key} \approx (19 + 12\log_2 N)\, N^2 C \tag{5.6c}$$

General expression for linear interpolator on keystone grid:

$$C_{linear}^{key} = (I_a + 3I_aI_r + N + 15N^2 + 2N^2\log_2 N)C_{m/d} \tag{5.7a}$$

$$+ (I_a + I_aI_r + N + 14N^2 + 2N^2\log_2 N)C_{a/s}$$

$$+ (2I_a + N^2)C_{trig}$$

Intermediate expression:

$$C_{linear}^{key} \approx (18 + 2\log_2 N)N^2C_{m/d} + (15 + 2\log_2 N)N^2C_{a/s} + N^2C_{trig} \quad (5.7b)$$

Simplified expression:

$$C_{linear}^{key} \approx (106 + 12\log_2 N)N^2C \quad (5.7c)$$

General expression for weighted sinc on keystone grid:

$$C_{wsinc}^{key} = (N + 2NI_a + 3N^2 + 2N^2K + 2N^2\log_2 N)C_{m/d} \quad (5.8a)$$
$$+ (N + NI_a + 6N^2 + 3N^2K + 2N^2\log_2 N)C_{a/s}$$
$$+ (NI_a + 2N^2)C_{trig} + N^2KC_{sinc}$$

Intermediate expression:

$$C_{wsinc}^{key} \approx (5 + 2\log_2 N + 2K)N^2C_{m/d} + (7 + 2\log_2 N + 2K)N^2C_{a/s} \quad (5.8b)$$
$$+ 3N^2C_{trig} + N^2KC_{sinc}$$

Simplified expression:

$$C_{wsinc}^{key} \approx (35 + 12\log_2 N + 13K)N^2C \quad (5.8c)$$

Using Eqs. (5.6c), (5.7c), and (5.8c) with N=1024, and K=12, it is found that $C_{NN}^{key} \approx 0.91\, C_{NN}$, $C_{linear}^{key} \approx 0.72\, C_{ID}^2$, and $C_{w-sinc}^{key} \approx 0.57\, C_{w-sinc}$. The complexity of the NN on the keystone grid has not improved much over the polar grid. This is because the FFT complexity is dominating both expressions. The w-sinc interpolator, which is dominated by the interpolation step, shows a noticeable cost improvement, owing to the removal of the range line interpolations.

General expression for the cubic spline on keystone grid:

$$C_{spline}^{key} = (N + 3N^2 + 8N^2 + 2N^2\log_2 N)C_{m/d} \quad (5.9a)$$
$$+ (6N + 27NI_a + 8N^2 + 2N^2\log_2 N)C_{a/s}$$
$$+ I_a\, C_{trig}$$

Intermediate expression:

$$C_{spline}^{key} \approx (11 + 2\log_2 N)N^2C_{m/d} + (35 + 2\log_2 N)N^2C_{a/s} \quad (5.9b)$$

Simplified expression:

$$C_{spline}^{key} \approx (90 + 12\log 2N)N^2C \tag{5.9c}$$

General expression for the B-splines on keystone grid:

$$C_{B-spline}^{key} = (N + 2Nl_a + 3N^2 + 12N^2 + 2N^2\log_2 N)C_{m/d} \tag{5.10a}$$
$$+ (N + Nl_a + 6N^2 + 10N^2 + 2N^2\log_2 N)C_{a/s}$$
$$+ NC_{trig}$$

Intermediate expression:

$$C_{B-spline}^{key} \approx (17 + 2\log_2 N)N^2C_{m/d} + (17 + 2\log_2 N)N^2C_{a/s} \tag{5.10b}$$
$$+ NC_{trig}$$

Simplified expression:

$$C_{B-spline}^{key} \approx [\,(102 + 12\log_2 N)N^2 + N\,]C \tag{5.10c}$$

General expression for chirp-z algorithm on keystone grid:

$$C_{CZT} = (4l_a + 19l_aN + 12l_aN\log_2 N + l_a(N/a)\log_2 N)C_{m/d} \tag{5.11a}$$
$$+ (l_a + 64l_aN + 12l_aN\log_2 N + l_aN\log_2 N)C_{a/s}$$

Intermediate expression:

$$C_{CZT} \approx (41 + 13\log_2 N)N^2C_{m/d} + (72 + 13\log_2 N)N^2C_{a/s} \tag{5.11b}$$

Simplified expression:

$$C_{CZT} \approx (277 + 76\log_2 N)\,N^2C \tag{5.11c}$$

For a 1024 by 1024 input grid, the chirp-z algorithm has a complexity measure of $1037N^2C$ which is comparable to a polar grid w-sinc interpolator with K=29 ($1052N^2C$) and a keystone w-sinc interpolator with K=68 (1039). This apparently high complexity measure is chiefly due to the large number of complex additions and multiplies in the chirp-z algorithm and the increase in array size from the extensive zero padding.

5.8 Summary

This chapter has presented several evaluations of the interpolators described and analyzed in Chapter 4. The required interpolator accuracy is dependent upon the acceptable reconstruction accuracy, measured here as a multiplicative-noise-ratio, and on the subarray location within the toroidal slice. The more radially distant subarrays require a higher order algorithm to reduce the MNR, sidelobes, and spurious targets, while the subarrays closer to the origin are nearly rectangular and require relatively little computation to produce a good reconstruction.

CHAPTER 6

CONCLUSIONS AND FURTHER RESEARCH

The problem of SAR data interpolation in the Fourier domain was examined, and several types of interpolator algorithms were discussed, analyzed, and evaluated experimentally. A true analysis of the polar to rectangular interpolation problem seems intractable, so the problem was reduced to a rectangular-rectangular problem and finally a one-dimensional problem of interpolation in the Fourier domain. Originally, much work was done to simulate an exact target response, but this obscured the effects that were being studied. The entire SAR and tomographic grid interpolation problem is that of finding an interpolator that is reasonably fast and yet has a transform that is close to an ideal low-pass filter. The various two-dimensional interpolators that were in current use [21] were more carefully studied, and some new interpolators for Fourier space were proposed for the SAR problem, i.e., inverse distance squared, windowed sinc, and cubic splines.

The inverse distance squared interpolator has not been seen in the recent DSP literature, but proved to generate good reconstruction at a lower cost than the more heavily used inverse distance algorithm. All of the known examples which examine DSP interpolators compare a *new* kernel with a sinc or truncated sinc. It seems that this comparison is unfair, since the windowed sinc has a much better response and is not too difficult to compute. Spline interpolation has also been successfully used in the area of Fourier domain reconstruction. This work helps sort out the meaning of *spline* in the current literature.

It is unlikely that an interpolation kernel will be discovered which is both easy to implement (fast) and has the desired spectrum (ideal low-pass filter) since the two criteria work against each other. Classical optimization procedures are ineffective here because the algorithm cost function is too difficult to parametize well. The weighted sinc is the best example of an interpolation algorithm whose order can be adjusted to reduce interpolation error below a given specification, though at the expense of processing time. The optimal Fourier domain interpolator

is global in nature, taking all the known data into account. However, this usually leads to prohibitive memory usage and computation times. Better methods exist for producing better image quality, such as iterative techniques, but these are far and away too expensive for the type of real-time processing that is desired.

A new method of evaluating the windowed point target response was presented as an alternative to the MSE criterion often used in image processing - the MNR and MNR/CPU ratio. These figures of merit proved useful in rating the interpolator image quality and in making algorithm comparisons.

A novel approach to the interpolation-inversion stage was presented via the chirp-z transform. Although it did not seem competitive with the weighted sinc interpolator, it has promise as a good alternative.

The alternative sampling grids reduced the interpolation error dramatically and were much faster to implement, due to the one-dimensional nature of the reconstruction. It is suggested that these raster designs be used in actual hardware designs due to the tremendous CPU cost savings they offer.

In short, the algorithm which produced the worst reconstruction was the nearest neighbor, followed by the inverse distance squared, and then inverse distance. These algorithms were not competitive with the separable interpolators: the weighted sinc and cubic splines. The weighted sinc had the advantage of having an adjustable parameter (order) which could be increased to improve reconstruction to acceptable levels, while the cubic spline was easier to implement for the same level of image quality. The chirp z-transform produced a good reconstructed image, yet was not very cost competitive with the separable algorithms due to the extended FFT size in the convolution stage.

6.1 Further Work

The advent of digital processing of SAR signals has opened a whole area of research. The interpolation problem examined in this work is useful in the direct inversion of the Fourier data set; however, there are other inversion algorithms which may lead to faster reconstructions *if* they in turn are made faster, specifically, convolutional back-projection. The work here has also led to many other questions concerning 2D interpolation and SAR. Additional research topics are presented below.

6.1.1 Full data array evaluations

The computer evaluations in this work were done on a relatively small data set. This had the advantage that the effects of the interpolator in different Fourier regions could be studied. However, an actual implementation of a 1024 by 1024 (or larger) data array would generate an image which is the coherent sum of all the small sub-arrays. It would be very useful to work with the large polar grid interpolation problem and examine the results for any new phenomena which may appear. The use of the newer generation supercomputers seems ideal for this higher order problem.

6.1.2 Oversampling

As mentioned in Chapter 4, the true input data of a working system is sampled at a rate much higher than required for the given system resolution. Work should be done to see how this higher volume of data, and hence finer sample spacing, could be used prior to prefiltering to reduce interpolation error.

6.1.3 Spline approximation to the sinc

Splines have been increasingly popular in DSP. Though the usual Fourier domain interpretation is lacking, they are easy to evaluate and have some nice mathematical properties. Since it was shown that the weighted sinc provided the best reconstruction, it may be useful to see if a

cubic spline approximation to the sinc would prove to be as accurate.

6.1.4 Spatially varying interpolation order

It was seen that different parts of the Fourier data set require different degrees of interpolation accuracy based on the sample spacing and angular orientation to the output grid. It is suggested that the interpolator order be made spatially dependent to minimize the amount of computation required, i.e., a low-order sinc could be used in the (1,1) region of the torus and a higher order interpolator could be used in the more (radially) distant parts of the polar grid (16,8). Empirically, it seems that the required interpolator order is more dependent on sample rate changes, resulting in spurious targets, than the degree of grid rotation.

The order of the separable sinc interpolator studied here was always the same in both radial and azimuthal directions. This may not be needed, as the azimuthal rate change demands a higher order interpolator than the range lines. Further computational savings may result if the range and azimuth interpolator orders are minimized to correspond to a prescribed accuracy for each dimension.

APPENDIX A

A FAST EVALUATION OF A PERIODICALLY SAMPLED SINE

A major obstacle in the use of the weighted sinc kernel is the cost of evaluating the two transcendentals within the kernel function:

$$h(x) = \frac{(.54+.46\cos(x*c_2))*\sin(x*c_1)}{x*c_1} \qquad (A.1)$$

where $x/c_1 = \pi$ and c_2 determines the extent of the Hamming window (the interpolator order).

This can be approached in many ways. First is the direct evaluation method. The interpolation kernel g(x) is a simple evaluation using floating point sine and cosine libraries. While this produces very accurate results, it is the most time-consuming, because the transcendentals are usually computed with a power series expansion with many terms.

An alternative to the direct method is with a table lookup. Here, the sine function is stored as a finely sampled array stored in a ROM, and the values of the transcendentals are calculated by finding the closest value in the table. A finer evaluation may be obtained by linearly interpolating between the two nearest table entries. This, however, can produce additional error in the kernel. Also, the error in the sinc function greatly increases as x approaches zero. This is, of course, due to sin(x) and x approaching zero at the same rate. Machine precision begins to cause errors in the quotient. To correct this, we can store the sinc(x) as one table and the weighted cos(x) as another. Note, too, that for storage savings, only the positive half of sinc(x) needs to be stored, and only one-fourth of the cosine function must be tabled. Half-angle formulas may be used to index to the correct table value.

If the data are equally spaced, a third, recursive, method may be used. Because we are computing a sinusoid at constantly spaced points, we can take advantage of the complex exponential to recursively yield successive sine values. We wish to calculate $\sin(n*\theta + \alpha)$ with n being the order of the interpolator. θ and α are determined by the position of the kernel function relative to the known data. We know that

$$\text{Imag}\{e^{j\alpha}\} = \sin(\alpha) \tag{A.2}$$

Using this identity, we can compute

$$S_n = \sin(n*\theta + \alpha) \qquad \text{For } n = 0,\ldots..N-1 \tag{A.3}$$

$$\sin(n*\theta + \alpha) = \text{Imag}\{e^{j(n*\theta + \alpha)}\} \tag{A.4a}$$

$$= \text{Imag}\{e^{j\theta n}e^{j\alpha}\} \tag{A.4b}$$

Let

$$\xi = e^{j\theta} \tag{A.5}$$

and

$$\sigma_0 = e^{j\alpha} \tag{A.6}$$

Then σ_i is recursively defined as

$$\sigma_i = \sigma_{i-1} * \xi \qquad i = 1\ldots..N-1 \tag{A.7}$$

Then:

$$S_i = \text{Imag}\{\theta_i\} \qquad i = 1\ldots..N-1 \tag{A.8}$$

This is a significant computational improvement over the standard power series expansion which is computed for each evaluation. The cosine term of A.1 may be computed in the same manner.

APPENDIX B

PLACING THE DC POINT IN THE IMAGE CENTER

The final stage of the SAR processing includes a circular permutation of the 2D data set to place the dc point in the center. Since this is actually the spatial domain, this corresponds to the terrain center. This step may also be performed in the Fourier domain with a slight computational savings.

As shown by Rosenfeld and Kak [74] the output of the FFT will be rotated by one-half the image size in both coordinates if it is first multiplied in the frequency domain by $(-1)^{m+n}$. That is, given the original Fourier data array F(m,n), define $\hat{F}(m,n)$:

$$\hat{F}(m,n) = F(m,n)\cdot(-1)^{m+n} \tag{B.1}$$

then the inverse transform of $\hat{F}(m,n)$ becomes

$$\hat{f}(j,k) = \sum_{j=0}^{M-1}\sum_{k=0}^{N-1} F(m,n)\cdot(-1)^{m+n}\cdot e^{j2\pi\left|\frac{jm}{M}+\frac{kn}{N}\right|} \tag{B.2}$$

but we can rewrite $(-1)^{m+n}$ as

$$(-1)^{m+n} = e^{-j2\pi\left(\frac{m+n}{2}\right)} \tag{B.3a}$$

$$= e^{-j2\pi\,(m/2 + n/2)} \tag{B.3b}$$

$$= e^{-j2\pi\left|\frac{(M/2)m}{M}+\frac{(N/2)n}{N}\right|} \tag{B.3c}$$

substituting (B.3c) into (B.2) and gathering terms yields

$$\hat{f}(j,k) = \sum_{j=0}^{M-1}\sum_{k=0}^{N-1} F(m,n)\cdot e^{j2\pi\left|\frac{(j-M/2)m}{M}+\frac{(k-N/2)n}{N}\right|} \tag{B.4}$$

$$= f(j-M/2,\, k-N/2) \quad \text{(QED)}$$

APPENDIX C

EVALUATION RESULTS FOR THREE GRID FORMATS

TABLE C.1. Evaluation Results for Fourier Piece (2,8) on Polar Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-5.64	3.28	1.72
-23,24	G-ID	(0,1)	-14.56	19.82	0.73
-23,24	G-ID	(0,2)	-18.41	5.75	3.20
-23,24	G-ID	(0,3)	-17.00	22.57	0.75
-23,24	G-ID	(0,4)	-14.94	21.17	0.71
-23,24	G-ID	(0,5)	-13.37	19.37	0.69
-23,24	wsinc	2	-18.92	17.7	1.07
-23,24	wsinc	4	-24.50	28.4	0.86
-23,24	wsinc	6	-30.37	41.3	0.74
-23,24	wsinc	8	-36.53	53.0	0.69
-23,24	wsinc	10	-42.65	61.7	0.69
-23,24	wsinc	12	-46.88	70.6	0.66
-23,24	wsinc	14	-48.06	82.7	0.47
-23,24	wsinc	16	-48.24	94.7	0.51
-23,24	wsinc	18	-48.27	102.5	0.47
-23,24	wsinc	20	-48.27	112.9	0.43
-23,24	wsinc	22	-48.30	126.0	0.38
-23,24	wsinc	24	-48.38	134.0	0.36
-23,24	B-spline	-0.25	-23.01	17.22	1.34
-23,24	B-spline	-0.50	-25.17	17.22	1.46
-23,24	B-spline	-0.75	-27.62	17.22	1.60
-23,24	B-spline	-1.00	-30.46	17.22	1.77
-23,24	Spline (IMSL)		-32.41	27.7	1.17

TABLE C.2. Evaluation Results for Fourier Piece (10,5) on Polar Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-5.31	3.28	1.62
-23,24	G-ID	(0,1)	-8.83	19.82	0.45
-23,24	G-ID	(0,2)	-9.02	5.75	1.57
-23,24	G-ID	(0,3)	-8.78	22.57	0.39
-23,24	G-ID	(0,4)	-8.48	21.17	0.40
-23,24	G-ID	(0,5)	-8.17	19.37	0.42
-23,24	wsinc	2	-9.27	17.7	0.52
-23,24	wsinc	4	-14.42	28.4	0.51
-23,24	wsinc	6	-19.57	41.3	0.47
-23,24	wsinc	8	-24.65	53.0	0.47
-23,24	wsinc	10	-29.75	61.7	0.48
-23,24	wsinc	12	-35.10	70.6	0.50
-23,24	wsinc	14	-40.82	82.7	0.49
-23,24	wsinc	16	-45.88	94.7	0.48
-23,24	wsinc	18	-47.85	102.5	0.47
-23,24	wsinc	20	47.99	112.9	0.43
-23,24	wsinc	22	-47.96	126.0	0.38
-23,24	wsinc	24	-47.96	134.0	0.36
-23,24	B-spline	-0.25	-14.10	17.22	0.82
-23,24	B-spline	-0.50	-16.30	17.22	0.95
-23,24	B-spline	-0.75	-18.70	17.22	1.09
-23,24	B-spline	-1.00	-21.40	17.22	1.18
-23,24	Spline (IMSL)		-24.19	27.7	0.87

TABLE C.3. Evaluation Results for Fourier Piece (16,1) on Polar Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	|MNR/CPU|
-23,24	NN		-1.97	3.28	0.60
-23,24	G-ID	(0,1)	-4.41	19.82	0.22
-23,24	G-ID	(0,2)	-4.45	5.75	0.77
-23,24	G-ID	(0,3)	-4.30	22.57	0.19
-23,24	G-ID	(0,4)	-3.95	21.17	0.19
-23,24	G-ID	(0,5)	-3.66	19.37	0.19
-23,24	wsinc	2	-4.13	17.7	0.23
-23,24	wsinc	4	-7.45	28.4	0.26
-23,24	wsinc	6	-10.62	41.3	0.26
-23,24	wsinc	8	-13.76	53.0	0.26
-23,24	wsinc	10	-17.01	61.7	0.28
-23,24	wsinc	12	-20.53	70.6	0.29
-23,24	wsinc	14	-24.48	82.7	0.30
-23,24	wsinc	16	-29.10	94.7	0.31
-23,24	wsinc	18	-34.66	102.5	0.34
-23,24	wsinc	20	-41.35	112.0	0.37
-23,24	wsinc	22	-47.10	126.0	0.37
-23,24	wsinc	24	-47.91	134.0	0.36
-23,24	B-spline	-0.25	-7.67	17.22	0.45
-23,24	B-spline	-0.50	-9.29	17.22	0.54
-23,24	B-spline	-0.75	-11.06	17.22	0.64
-23,24	B-spline	-1.00	-13.07	17.22	0.76
-23,24	Spline (IMSL)		-15.35	27.7	0.55

TABLE C.4. Evaluation Results for Fourier Piece (16,8) on Polar Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23.24	NN		-0.97	3.28	0.30
-23.24	G-ID	(0,1)	-6.18	19.82	0.31
-23.24	G-ID	(0,2)	-4.93	5.75	0.86
-23.24	G-ID	(0,3)	-4.50	22.57	0.20
-23.24	G-ID	(0,4)	-4.24	21.17	0.20
-23.24	G-ID	(0,5)	-4.01	19.37	0.21
-23.24	wsinc	2	-3.99	17.7	0.23
-23.24	wsinc	4	-7.56	28.4	0.27
-23.24	wsinc	6	-10.74	41.3	0.26
-23.24	wsinc	8	-13.80	53.0	0.26
-23.24	wsinc	10	-16.97	61.7	0.28
-23.24	wsinc	12	-20.42	70.6	0.29
-23.24	wsinc	14	-24.36	82.7	0.29
-23.24	wsinc	16	-29.03	94.7	0.31
-23.24	wsinc	18	-34.75	102.5	0.34
-23.24	wsinc	20	-41.90	112.9	0.37
-23.24	wsinc	22	-47.88	126.0	0.38
-23.24	wsinc	24	-47.44	134.0	0.35
-23.24	B-spline	-0.25	-7.19	17.22	0.42
-23.24	B-spline	-0.50	-8.67	17.22	0.50
-23.24	B-spline	-0.75	-10.26	17.22	0.60
-23.24	B-spline	-1.00	-12.01	17.22	0.70
-23.24	Spline (IMSL)		-14.72	27.7	0.54

TABLE C.5. Evaluation Results for Fourier Piece (2,8) on Equi-PRF grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-5.64	3.58	1.58
-23,24	G-ID	(0,1)	-14.67	19.02	0.77
-23,24	G-ID	(0,2)	-18.61	5.67	3.28
-23,24	G-ID	(0,3)	-17.12	18.75	0.91
-23,24	G-ID	(1,1)	-10.38	63.62	0.16
-23,24	G-ID	(1,2)	-17.15	11.13	1.54
-23,24	G-ID	(1,3)	-17.36	70.37	0.25
-23,24	G-ID	(2,1)	-8.18	151.82	0.05
-23,24	G-ID	(2,2)	-16.24	30.48	0.53
-23,24	G-ID	(2,3)	-17.44	207.55	0.08
-23,24	wsinc	2	-19.10	11.98	1.59
-23,24	wsinc	4	-24.58	22.57	1.09
-23,24	wsinc	6	-30.40	32.70	0.93
-23,24	wsinc	8	-36.54	41.10	0.89
-23,24	wsinc	10	42.64	50.73	0.84
-23,24	wsinc	12	-46.80	59.7	0.78
-23,24	wsinc	14	-48.06	69.53	0.69
-23,24	wsinc	16	-48.16	79.20	0.61
-23,24	wsinc	18	-48.19	88.93	0.54
-23,24	wsinc	20	-48.22	99.03	0.49
-23,24	wsinc	22	-48.19	110.03	0.44
-23,24	wsinc	24	-48.24	122.02	0.40
-23,24	B-spline	-0.25	-23.84	22.13	1.08
-23,24	B-spline	-0.50	-26.00	22.13	1.17
-23,24	B-spline	-0.75	-28.03	22.13	1.27
-23,24	B-spline	-1.00	-31.17	22.13	1.41
-23,24	Spline (IMSL)		-29.63	26.75	1.11

TABLE C.6. Evaluation Results for Fourier Piece (10,5) on Equi-PRF grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23.24	NN		-5.56	3.58	1.55
-23.24	G-ID	(0,1)	-8.89	19.02	0.47
-23.24	G-ID	(0,2)	-9.18	5.67	1.62
-23.24	G-ID	(0,3)	-9.00	18.75	0.48
-23.24	G-ID	(1,1)	-7.44	63.62	0.12
-23.24	G-ID	(1,2)	-8.40	17.15	0.49
-23.24	G-ID	(1,3)	-8.73	70.37	0.12
-23.24	G-ID	(2,1)	-7.37	151.82	0.049
-23.24	G-ID	(2,2)	-8.20	20.48	0.40
-23.24	G-ID	(2,3)	-8.68	207.55	0.042
-23.24	wsinc	2	-9.44	11.98	0.79
-23.24	wsinc	4	-14.52	22.57	0.64
-23.24	wsinc	6	-19.64	32.70	0.60
-23.24	wsinc	8	-24.72	41.10	0.60
-23.24	wsinc	10	-29.83	50.73	0.59
-23.24	wsinc	12	-35.22	59.78	0.59
-23.24	wsinc	14	-41.04	69.53	0.59
-23.24	wsinc	16	-46.15	79.20	0.58
-23.24	wsinc	18	-47.96	88.93	0.54
-23.24	wsinc	20	-48.02	99.03	0.48
-23.24	wsinc	22	-48.00	110.03	0.44
-23.24	wsinc	24	-48.04	122.02	0.39
-23.24	B-spline	-0.25	-15.23	22.13	0.69
-23.24	B-spline	-0.50	-17.12	22.13	0.77
-23.24	B-spline	-0.75	-19.63	22.13	0.89
-23.24	B-spline	-1.00	-22.99	22.13	1.04
-23.24	Spline (IMSL)		-37.60	23.82	1.58

TABLE C.7. Evaluation Results for Fourier Piece (16,1) on Equi-PRF grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-1.96	3.58	0.55
-23,24	G-ID	(0,1)	-4.38	19.02	0.23
-23,24	G-ID	(0,2)	-4.52	5.67	0.80
-23,24	G-ID	(0,3)	-4.27	18.75	0.23
-23,24	G-ID	(1,1)	-1.40	63.62	0.022
-23,24	G-ID	(1,2)	-3.26	17.15	0.19
-23,24	G-ID	(1,3)	-3.89	70.37	0.055
-23,24	G-ID	(2,1)	-0.15	151.82	0.001
-23,24	G-ID	(2,2)	-2.88	20.48	0.141
-23,24	G-ID	(2,3)	-3.82	207.55	0.018
-23,24	wsinc	2	-4.10	11.98	0.342
-23,24	wsinc	4	-7.42	22.57	0.329
-23,24	wsinc	6	-10.58	32.70	0.324
-23,24	wsinc	8	-13.72	41.10	0.334
-23,24	wsinc	10	-16.97	50.73	0.335
-23,24	wsinc	12	-20.47	59.78	0.342
-23,24	wsinc	14	-24.42	69.53	0.351
-23,24	wsinc	16	-29.03	79.20	0.367
-23,24	wsinc	18	-34.60	88.93	0.389
-23,24	wsinc	20	-41.31	99.03	0.417
-23,24	wsinc	22	-47.00	110.03	0.427
-23,24	wsinc	24	-47.53	122.02	0.390
-23,24	B-spline	-0.25	-8.89	22.13	0.04
-23,24	B-spline	-0.50	-10.23	22.13	0.46
-23,24	B-spline	-0.75	-12.04	22.13	0.54
-23,24	B-spline	-1.00	-14.88	22.13	0.67
-23,24	Spline (IMSL)		-16.43	26.75	0.614

TABLE C.8. Evaluation Results for Fourier Piece (16,8) on Equi-PRF grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	|MNR/CPU|
-23,24	NN		-1.00	3.58	0.28
-23,24	G-ID	(0,1)	-6.21	19.02	0.326
-23,24	G-ID	(0,2)	-4.97	5.67	0.88
-23,24	G-ID	(0,3)	-4.55	18.75	0.24
-23,24	G-ID	(1,1)	-2.98	63.62	0.047
-23,24	G-ID	(1,2)	-3.38	17.15	0.20
-23,24	G-ID	(1,3)	-3.90	70.37	0.055
-23,24	G-ID	(2,1)	-1.28	151.82	0.008
-23,24	G-ID	(2,2)	-2.95	20.48	0.144
-23,24	G-ID	(2,3)	-3.79	207.55	0.018
-23,24	wsinc	2	-4.04	11.98	0.337
-23,24	wsinc	4	-7.62	22.57	0.338
-23,24	wsinc	6	-10.81	32.70	0.331
-23,24	wsinc	8	-13.89	41.10	0.338
-23,24	wsinc	10	-17.07	50.73	0.336
-23,24	wsinc	12	-20.55	59.78	0.344
-23,24	wsinc	14	-24.52	69.53	0.353
-23,24	wsinc	16	-29.24	79.20	0.369
-23,24	wsinc	18	-35.04	88.93	0.394
-23,24	wsinc	20	-42.30	99.03	0.427
-23,24	wsinc	22	-48.06	110.03	0.437
-23,24	wsinc	24	-47.44	122.02	0.389
-23,24	B-spline	-0.25	-8.32	22.13	0.38
-23,24	B-spline	-0.50	-9.03	22.13	0.41
-23,24	B-spline	-0.75	-11.83	22.13	0.54
-23,24	B-spline	-1.00	-14.33	22.13	0.65
-23,24	Spline (IMSL)		-42.57	26.75	1.59

TABLE C.9. Evaluation Results for Fourier Piece (2,8) on Keystone Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	|MNR/CPU|
-23.24	NN		-7.26	0.67	10.83
-23.24	G-ID	1	-24.16	0.85	28.42
-23.24	G-ID	2	-22.23	1.42	15.66
-23.24	G-ID	3	-17.66	1.65	10.70
-23.24	G-ID	4	-15.15	1.47	10.31
-23.24	wsinc	2	-22.60	4.63	4.88
-23.24	wsinc	4	-26.95	8.70	3.10
-23.24	wsinc	6	-31.86	13.2	2.41
-23.24	wsinc	8	-37.43	17.3	2.16
-23.24	wsinc	10	-43.14	21.90	1.97
-23.24	wsinc	12	-46.97	26.00	1.81
-23.24	wsinc	14	-48.04	29.85	1.61
-23.24	wsinc	16	-48.12	34.52	1.39
-23.24	wsinc	18	-48.14	37.57	1.28
-23.24	Chirp-Z		-1.32	26.50	0.05
-23.24	B-spline	-0.25	-34.12	7.95	4.29
-23.24	B-spline	-0.50	-35.88	7.95	4.51
-23.24	B-spline	-0.75	-36.12	7.95	4.54
-23.24	B-spline	-1.00	-36.98	7.95	4.65
-23.24	Spline (IMSL)		-34.80	12.06	2.89

TABLE C.10. Evaluation Results for Fourier Piece (16,1) on Keystone Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-2.01	0.67	3.00
-23,24	G-ID	1	-7.68	0.85	9.04
-23,24	G-ID	2	-5.56	1.42	3.92
-23,24	G-ID	2.5	-5.03	6.68	0.75
-23,24	G-ID	3	-4.64	1.65	2.81
-23,24	wsinc	2	-4.36	4.63	0.942
-23,24	wsinc	4	-8.26	8.70	0.949
-23,24	wsinc	6	-12.38	13.2	0.938
-23,24	wsinc	8	-16.91	17.3	0.977
-23,24	wsinc	10	-22.05	21.90	1.007
-23,24	wsinc	12	-28.10	26.0	1.081
-23,24	wsinc	14	-35.47	29.85	1.188
-23,24	wsinc	16	-44.05	34.65	1.271
-23,24	wsinc	18	-48.25	37.57	1.284
-23,24	Chirp-Z		-30.92	26.50	1.17
-23,24	B-spline	-0.25	-15.53	7.95	1.95
-23,24	B-spline	-0.50	-16.43	7.95	2.07
-23,24	B-spline	-0.75	-16.89	7.95	2.12
-23,24	B-spline	-1.00	-17.35	7.95	2.18
-23,24	Spline (IMSL)		-15.36	12.06	1.27

TABLE C.11. Evaluation Results for Fourier Piece (10,5) on Keystone Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23,24	NN		-6.80	0.67	10.15
-23,24	G-ID	1	-17.39	0.85	20.46
-23,24	G-ID	2	-15.60	1.42	10.99
-23,24	G-ID	3	-13.71	1.65	8.31
-23,24	G-ID	4	-12.36	1.47	8.41
-23,24	wsinc	2	-14.87	4.63	3.21
-23,24	wsinc	4	-18.54	8.70	2.13
-23,24	wsinc	6	-22.86	13.2	1.71
-23,24	wsinc	8	-27.85	17.3	1.61
-23,24	wsinc	10	-33.63	21.90	1.54
-23,24	wsinc	12	-40.25	26.0	1.55
-23,24	wsinc	14	-46.24	29.85	1.55
-23,24	wsinc	16	-48.21	34.52	1.40
-23,24	wsinc	18	-48.32	37.57	1.29
-23,24	Chirp-Z		-8.62	26.50	0.325
-23,24	B-spline	-0.25	-16.43	7.95	2.07
-23,24	B-spline	-0.50	-17.33	7.95	2.18
-23,24	B-spline	-0.75	-17.93	7.95	2.26
-23,24	B-spline	-1.00	-18.55	7.95	2.33
-23,24	Spline (IMSL)		-25.77	12.06	2.14

TABLE C.12. Evaluation Results for Fourier Piece (16,8) on Keystone Grid.

Target Position	Interp.	Parameter	MNR (dB)	CPU time (seconds)	| MNR/CPU |
-23.24	NN		-1.97	0.67	2.94
-23.24	G-ID	1	-7.71	0.85	9.07
-23.24	G-ID	2	-5.57	1.42	3.92
-23.24	G-ID	3	-4.64	1.65	2.81
-23.24	G-ID	4	-4.09	1.47	2.78
-23.24	wsinc	2	-4.37	4.63	0.94
-23.24	wsinc	4	-8.23	8.70	0.95
-23.24	wsinc	6	-12.41	13.2	0.94
-23.24	wsinc	8	-16.97	17.3	0.98
-23.24	wsinc	10	[illegible]	21.90	1.01
-23.24	wsinc	12	-28.33	26.0	1.09
-23.24	wsinc	14	-36.04	29.85	1.21
-23.24	wsinc	16	-45.20	34.52	1.31
-23.24	wsinc	18	-47.75	37.57	1.27
-23.24	Chirp-Z		-14.75	26.50	0.557
-23.24	B-spline	-0.25	-15.04	7.95	1.89
-23.24	B-spline	-0.50	-15.73	7.95	1.98
-23.24	B-spline	-0.75	-16.32	7.95	2.05
-23.24	B-spline	-1.00	-17.03	7.95	2.14
-23.24	Spline (IMSL)		-15.41	12.06	1.28

REFERENCES

[1] M. I. Skolnik, "Fifty years of radar," *Proc. of the IEEE*, vol. 73, pp. 182-197, Feb. 1985.

[2] J. J. Kovaly, *Synthetic Aperture Radar*, Dedham, MA: Artech House, Inc., 1976.

[3] R. O. Harger, *Synthetic Aperture Radar*, New York, NY: Academic Press, 1970.

[4] L. J. Cutrona, "Optical data processing and filtering systems," *IEEE Trans. Inform. Theory*, vol. IT-6, pp. 386-400, June 1960.

[5] H. Stark, "Sampling theorems in polar coordinates," *J. Opt. Soc. Amer.*, vol. 69, pp. 1519-1525, Nov. 1979.

[6] C. F. R. Weiman and G. Chaikin, "Logarithmic spiral grids for image processing and display," *Comp. Graph. Image Proc.*, vol. 11, pp. 197-226, Nov. 1979.

[7] D. C. Munson, Jr., J. D. O'Brien, and W. K. Jenkins, "A unified theory for high resolution image reconstruction in spotlight mode synthetic aperture radar and computer-aided tomography," *Proc. of the IEEE*, 1984.

[8] D. C. Munson and J. L. C. Sanz, "The importance of random phase for image reconstruction from frequency offset Fourier data," in *IEEE International Conference on Acoust., Speech, and Signal Processing*, San Diego, CA, Mar. 19-21, 1984.

[9] K. Tomiyasu, "Tutorial review of synthetic aperture radar (SAR) with applications to imaging of the ocean surface," *Proc. IEEE*, vol. 66, p. May, 1978.

[10] J. D. Johnson and L. D. Farmer, "Use of side-looking air-borne radar for sea-ice surveillance," *J. Geophys. Res.*, vol. 76, Mar. 1971.

[11] J. A. Parker, R. V. Kenyon, and D. E. Troxel, "Comparison of interpolating methods for image resampling," *IEEE Trans. on Med. Imag.*, vol. MI-2, pp. 31-39, Mar. 1983.

[12] Chin-Hwa Lee, "Restoring spline interpolation of CT images," *IEEE Trans. on Med. Imag.*, vol. MI-2, pp. 142-149, Sept. 1983.

[13] R. M. Mersereau, "Direct Fourier transform techniques in 3-D image reconstruction," in *Comput. Biol Med.*, vol. 6, Great Britain, pp. 247-258, 1976.

[14] D. A. Schwartz, "Analysis and Experimental Investigation of Three Synthetic Aperture Radar Formats," Masters thesis, University of Illinois, 1983.

[15] W. G. Weis and W. K. Jenkins, "Theory and simulation of digital FFT processors for spotlight mode synthetic aperture radar," Lockheed Missiles and Space Corporation, Report D566409, Dec. 1977.

[16] H. V. Hance, W. K. Jenkins, and D. C. Munson, Jr., "Polar format SAR resource reduction," Avionics Laboratory, Wright-Patterson Air Force Base, OH, Rep. under Contract F33615-83-C-1034, June 1985.

[17] J. R. Klauder, A. C. Price, and et. al. "The theory and design of chirp radars," *The Bell Syst. Tech. J.*, vol. 39, pp. 745-808, July 1960.

[18] M. Bernfeld, "CHIRP Doppler radar," *Proc. of the IEEE*, vol. 72, pp. 540-541, Apr. 1984.

[19] M. E. Davidson, "The ill-conditioned nature of the limited angle tomography problem," in *Proc. IEEE International Conference on Circuits and Systems*, Montreal, Canada, May 1984.

[20] W. Arens, "Azimuth correlator for synthetic aperture radar," *NASA Tech Briefs*, Spring 1979.

[21] S. X. Pan and A. C. Kak, "A computational study of reconstruction algorithms for diffraction tomography: Interpolation versus filtered backprojection," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-31, Oct., 1983.

[22] H. Stark, J. W. Woods, I. Paul, and R. Hingorani, "An investigation of computerized tomography by direct Fourier inversion and optimum interpolation," *IEEE Trans. on Bio-Med. Eng.*, vol. BME-28, pp. 496-505, July 1981.

[23] H. Fan and J. L. C. Sanz, "Comments on direct Fourier reconstruction in computer tomography," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-33, pp. 446-448, Apr. 1985.

[24] R. B. Heffernan and R. H. T. Bates, "Image reconstruction from projections vi: Comparison of interpolation methods," *Optik*, vol. 50, pp. 19-33, 1978.

[25] M. D. Desai, "A New Method of Synthetic Aperture Radar Image Reconstruction Using Modified Convolution Back-Projection Algorithm," Ph.D thesis, University of Illinois, 1985.

[26] M. D. Desai and W. K. Jenkins, "Convolution back-projection image reconstruction for synthetic aperture radar imaging," in *Proc. of the Intern. Symposium on Circuits and Systems*, Montreal, Canada, pp. 261-264, May 1984.

[27] W. E. Higgins, "Tomographic Image Reconstruction via a Hankel transform approach and a low-noise digital filter structure," Ph.D thesis, University of Illinois, 1984.

[28] W. E. Higgins and D. C. Munson, Jr., "Tomographic image reconstruction via a Hankel transform approach," in *Seventeenth Asilomar Conference on Circuits, Systems, and Computers*, Pacific Grove, CA, Oct 31-Nov 2, 1983.

[29] G. H. Hostetter, "Recursive discrete Fourier transformation with unevenly spaced data," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-31, pp. 206-209, Feb. 1983.

[30] G. H. Hostetter, *Recursive spectral observation for fast transforms with unevenly spaced data*, Publication of University of California Electrical Engineering Department..

[31] J. C. Kirk, "A discussion of digital processing in synthetic aperture radar," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-11, pp. 367-337, May 1975.

[32] W. E. Brown, G. G. Houser, and R. E. Jenkins, "Synthetic aperture processing with limited storage and presumming," *IEEE Trans. on Aerosp. Electron. Syst.*, vol. AES-9, pp. 166-174, Mar. 1973.

[33] D. A. Hayner, "The Missing Cone Problem in Computer Tomography and a Model for Interpolation in Synthetic Aperture Radar," Ph.D thesis, University of Illinois, 1983.

[34] T. J. Rivlin, *An Introduction to the Approximation of Functions*, New York, NY: Dover, 1969.

[35] R. E. Crochiere and L. R. Rabiner, "Optimum fir digital filter implementations for decimation Interpolation, and Narrow-Band Filtering," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 444-456, Oct., 1975.

[36] R. W. Schafer and L. R. Rabiner, "A digital signal processing approach to interpolation," *Proc. IEEE*, vol. 61, pp. 692-702, June, 1973.

[37] Digital Signal Processing Committee, *Programs for Digital Signal Processing*, New York, NY: IEEE Press, 1979.

[38] M. G. Bellanger, J. L. Daguet, and G. P. Lepagnol, "Interpolation, extrapolation, and reduction of computation speed in digital filters," *IEEE Trans Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 231-235, Aug 1974.

[39] G. A. Nelson, L. L. Pfeifer, and R. C. Wood, "High-speed octave band digital filtering," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 58-65, Mar. 1972.

[40] R. R. Shively, "On multistage FIR filters with decimation," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 353-357, Aug. 1975.

[41] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*, Englewood Cliffs, NJ: Prentice-Hall, 1983.

[42] L. R. Rabiner and R. E. Crochiere, "A novel implementation for narrow-band FIR digital filters," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-23, pp. 457-464, Oct. 1975.

[43] N. S. Vasanthavada, "A note on the computational efficiency of the FIR multistage implementation of the fractional sampling rate conversion," *IEEE Trans. on Acoust. Speech, Signal Processing*, vol. ASSP-33, pp. 475-477, Apr. 1985.

[44] T. A. Ramstad, "Digital methods for conversion between arbitrary sampling frequencies," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 577-591, June 1984.

[45] K. C. McGill and L. J. Dorfman, "High-resolution alignment of sampled waveforms," *IEEE Trans. on Bio-Med. Eng.*, vol. BME-31, pp. 462-468, June 1984.

[46] A. Papoulis, *The Fourier Integral and its Applications*, New York, NY: McGraw-Hill, 1962.

[47] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," *Proc. 1968 ACM National Conference*, pp. 517-523.

[48] H. Akima, "A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points," *ACM Trans. on Mathematical Software*, vol. 4, pp. 148-159, June 1978.

[49] F. J. Harris, "On the use of windows for harmonic analysis with the discrete Fourier transform," *Proc. of the IEEE*, vol. 66, pp. 51-83, Jan. 1978.

[50] T. N. E. Greville, *Theory and Applications of Spline Functions*, New York, NY: Academic Press, 1969.

[51] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh, *The Theory of Splines and Their Applications*, New York, NY: Academic Press, 1967.

[52] J. R. Rice, *Numerical Methods, Software, and Analysis*, New York, NY: McGraw-Hill.

[53] D. E. Knuth, *The TexBook*, Reading, MA: Addison-Wesley, 1970.

[54] Carl de Boor, *A Practical Guide to Splines*, New York, NY: Springer-Verlag, 1985.

[55] L. E. Ostrander, "The Fourier transform of spline function approximations to continuous data," *IEEE Trans. Audio Electroacoust.*, vol. AU-19, pp. 103-104, Mar. 1971.

[56] L. L. Horowitz, "The effects of spline interpolation on power spectral density," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-22, pp. 22-27.

[57] H. S. Hou and H. C. Andrews, "Cubic splines for image interpolation and digital filtering," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-26, Dec. 1978.

[58] R. G. Keys, "Cubic convolution interpolation for digital image processing," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 1153-1160, Dec. 1981.

[59] L. R. Rabiner, B. Gold, and C. A. McGonegal, "An approach to the approximation problem for nonrecursive digital filters," *IEEE Trans. Audio Electroacoust.*, vol. AU-18, pp. 83-106, June 1970.

[60] N. C. Geckinli and D. Yavuz, "Some novel windows and a concise tutorial comparison of window families," *IEEE Trans. on Acoust., Speech, Signal Processing*, vol. ASSP-26, pp. 501-507, Dec. 1978.

[61] T. S. Huang, "Two-dimensional windows," *IEEE Trans. Audio Electroacoust.*, vol. AU-20, pp. 88-89, Mar. 1972.

[62] J. L. Auterman and et. al., *Radar and Optics Division Environmental Research Institute of Michigan AFL-TR-75-105*, Dec. 1975.

[63] A. Papoulis, *Signal Analysis*. New York, NY: McGraw Hill, 1977.

[64] D. Slepian and H. O. Pollak, "Prolate spheroidal wave functions, Fourier analysis and uncertainty," *Bell Syst. Tech. J.*, vol. 40, pp. 43-64, Jan. 1961.

[65] D. Slepian, "Prolate spheroidal wave functions, Fourier analysis and uncertainty - IV: Extensions to many dimensions, generalized prolate spheroidal functions," *Bell Syst. Tech. J.*, vol. 63, pp. 3009-3057, Nov. 1964.

[66] D. Slepian, "Prolate spheroidal wave functions, Fourier analysis, and uncertainty - V: The discrete case," *Bell Syst. Tech. J.*, vol. 57, pp. 1371-1429, 1978.

[67] J. D. O'Brien, "A Consideration of Signal Processing for Spotlight Synthetic Aperture Radar " Ph.D thesis, University of Illinois, 1984.

[68] J. J. Staehling, "Two-Dimensional Windowing Methods in Digital Signal Processing," Masters thesis, University of Illinois, 1984.

[69] A. Papoulis, "A new algorithm in spectral analysis and band-limited extrapolation," *IEEE Trans. Comput. Syst.*, vol. 22, pp. 735-742, 1975.

[70] K. Abend, "Spectrum analysis and resolution enhancement by bandlimited extrapolation," *IEEE 1980 ICASSP*, pp. 603-606, Apr. 1908.

[71] R. M. Mersereau and A. V. Oppenheim, "Digital reconstructions of multidimensional signals from their projections," *Proc. IEEE*, vol. 62, pp. 1319-1338, Oct. 1974.

[72] L. R. Rabiner, R. W. Schafer, and C. M. Radar, "The chirp z-transform algorithm and its application," *The Bell Syst. Tech. J.*, pp. 1249-1292, May-June 1969.

[73] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Englewood, NJ: Prentice Hall, Inc., 1975.

[74] A. Rosenfeld and A. C. Kak, *Digital Picture Processing, Vol. 1*. New York, NY: Academic Press, 1982.